The Return of Unix Command-Line Kung Fu

@Hal_Pomeranz

Hal Pomeranz
Deer Run Associates
PO Box 50638
Eugene, OR 97405

hal@deer-run.com
(541)683-8680
(541)683-8681 (fax)
http://www.deer-run.com/

[I wish to thank everybody who's attended this presentation and given me suggestions for improving the content. I haven't been able to always get your name/email address to thank you explicitly in the course notes, but your contributions are appreciated by me and everybody who uses this course. --Hal]

Who Is Hal Pomeranz?

- Independent consultant
- SANS Faculty Fellow, "oldest" surviving SANS instructor
  - Author, track lead for Sec506: Linux/Unix Security
  - Instructor for SANS Forensics classes
- Did I mention the blogs?
  - commandlinekungfu.com (w/ Ed Skoudis and Tim Medin)
  - http://blogs.sans.org/computer-forensics/

Welcome! My name is Hal Pomeranz and I've been working with Unix systems professionally since 1987. By the way, when I say "Unix", I mean all Unix-like systems, including Linux. It's all rock'n'roll to me…

For the last 10 years my wife Laura and I have been running our own consulting practice (although she claims she's "not technical anymore", my wife was using Unix systems many years before I was and she's still a mean hand with the `vi` text editor). I also have the curious distinction of being the "oldest" current SANS Faculty member (in terms of longevity with the organization, not by age), having presented my first tutorial for SANS in 1994 and various other talks at SANS conferences from the early '90s. I'm currently the track lead and primary instructor for SANS' Unix Security certification track (aka SANS Sec506).

I've been active in the Unix community throughout my career and have served on the Boards of several different computing and system administration organizations, including BayLISA (San Francisco Bay Area), BBLISA (Boston), and USENIX. I was the last Technical Editor for *Sys Admin Magazine,* from Jan 2004 through Aug 2007 when the magazine ceased publication. I've also helped to develop many of the existing Unix security standards, including those from the Center for Internet Security (*http://www.CISecurity.org/*). I am also a recipient of the annual SAGE Outstanding Achievement Award for my teaching and leadership in the field of System Administration.

Simple Output Redirections

- Output one way, errors another:

```
make >/tmp/build.log 2>/tmp/build.errors
```

- Getting stderr out of your way:

```
strace vi ~/.bashrc 2>/tmp/strace-output
```

- Output and errors together to the bit bucket:

```
make distclean >/dev/null 2>&1
```

http://blog.commandlinekungfu.com/2009/06/episode-47-fun-with-output-redirection.html

Add /dev/tcp/… Goodness

- Command output to network:

```
df >/dev/tcp/foo.example.com/9999
```

- Or a simple port checker:

```
$ echo >/dev/tcp/127.0.0.1/22
$ echo >/dev/tcp/127.0.0.1/23
bash: connect: Connection refused
bash: /dev/tcp/127.0.0.1/23: Connection refused
```

http://blog.commandlinekungfu.com/2010/04/episode-89-lets-scan-us-some-ports.html

Cleaning Up That Last Example

- Why doesn't this work?

```
$ echo >/dev/tcp/127.0.0.1/23 2>/dev/null \
    && echo live || echo dead
bash: connect: Connection refused
bash: /dev/tcp/127.0.0.1/23: Connection refused
dead
```

- Need to do output redirection in a sub-shell:

```
$ (echo >/dev/tcp/127.0.0.1/23) 2>/dev/null \
    && echo live|| echo dead
dead
```

http://blog.commandlinekungfu.com/2010/04/episode-89-lets-scan-us-some-ports.html

Beware Local Optimizations

- Simple port scanner:

```
$ for ((i=1; $i < 1024; i++)); do
    (echo >/dev/tcp/127.0.0.1/$i) 2>/dev/null \
        && echo $i/tcp live;
done
22/tcp live
631/tcp live
```

- But this is faster:

```
$ for ((i=1; $i < 1024; i++)); do
     echo >/dev/tcp/127.0.0.1/$i \
        && echo $i/tcp live;
done 2>/dev/null
22/tcp live
631/tcp live
```

http://blog.commandlinekungfu.com/2010/04/episode-89-lets-scan-us-some-ports.html

Fun With FIFOs

- You want to capture command output with **`script`**
- **`script`** wants to write to a local file
- This is bad from a forensic perspective
- Use a FIFO instead!

```
# mkfifo /tmp/fifo
# cat </tmp/fifo >/dev/tcp/192.168.1.1/8001 &
[1] 3066
# script -f /tmp/fifo
Script started, file is /tmp/fifo
```

http://blogs.sans.org/computer-forensics/2009/07/29/tricking-the-script-command/

More Fun With FIFOs

- You have a large disk image file
- You want to dump both ASCII and Unicode strings
- You don't want to have to read the image twice
- Use tee command with FIFO:

```
# strings -a -t d -e l </tmp/fifo | \
    gzip > strings.uni.gz &
[1] 23281
# cat ntfs.img | tee /tmp/fifo | \
    strings -a -t d | gzip >strings.ascii.gz
```

http://blogs.sans.org/computer-forensics/2010/01/27/fun-with-fifos-part-ii-output-splitting/

Kill! Kill! Kill!

- Kill processes by name:

```
pkill sshd
```

- Or perhaps more selectively:

```
pkill -P 1 sshd
```

- Kill processes by user:

```
pkill -u hal
```

http://blog.commandlinekungfu.com/2009/04/episode-22-death-to-processes.html

And With a Little Help From **`lsof`**…

- Kill processes by port:

```
kill `lsof -t -i :22`
```

- Unmount that volume:

```
# umount /home
umount: /home: device is busy
# kill `lsof -t /home`
# umount /home
```

http://blog.commandlinekungfu.com/2009/11/episode-69-destroy-all-connections.html

Killing By Start Time Is *Hard*

- Timestamps on **`/proc`** aren't related to proc start time
- **`pkill`** can only kill oldest (**`-o`**) or newest (**`-n`**) proc
- **`lsof`** has no options to select process start time
- **`ps -eo pid,comm,start_time`** is useless
- How awful is this?

```
ps -eo pid,comm,etime | \
   tail -n +2 | sed 's/[-:]/ /g' | \
   awk '{print $1, $2, $6, $5, $4, $3}' | \
   awk '{print $1, $2,
         ($3 + $4 * 60 + $5 * 3600 + $6 * 86400)}'
```

http://blog.commandlinekungfu.com/2010/05/episode-94-date-with-death.html

Got the **touch**

- Use **touch** (as root) to manipulate timestamps at will:

```
# touch -t 201001010000 /tmp/testing
# stat /tmp/testing
  File: `/tmp/testing'
  Size: 0            Blocks: 0          IO Block: 4096
Device: fc01h/64513d Inode: 5603        Links: 1
Access: (0644/-rw-r--r--)  Uid: (0/root)   Gid: (0/root)
Access: 2010-01-01 00:00:00.000000000 -0800
Modify: 2010-01-01 00:00:00.000000000 -0800
Change: 2010-03-08 16:42:33.993133369 -0800
```

http://blog.commandlinekungfu.com/2009/04/episode-29-finding-time.html

http://blog.commandlinekungfu.com/2010/02/episode-80-time-bandits.html

Use Your `touch` for Good, Not Evil

- "find … -mtime …" only works on one-day intervals:

```
find / -mtime -7
```

- Combine touch with "find … -newer …" and do better:

```
touch -t 201003021337 /tmp/timestamp
find / -newer /tmp/timestamp
```

http://blog.commandlinekungfu.com/2009/04/episode-29-finding-time.html

Since We're Talking Timestamps…

- Sort directory entries by last modified time (-t):

```
ls -lrt
```

- Add **-u** option to sort by last access time

http://blog.commandlinekungfu.com/2010/01/not-ready-yet-episode-79-sort-of-list.html

What About ctime?

- Hacking ctime values generally requires specialized tool
- Enter **`debugfs`** on Linux EXT file systems:

```
# df -h /tmp/testing
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/elk-root  961M  665M  247M  73% /
# debugfs -w -R 'set_inode_field /tmp/testing
                 ctime 201001012222' /dev/mapper/elk-root
# debugfs -R 'stat /tmp/testing' /dev/mapper/elk-root
[…]
 ctime: 0x4b3ee608:ecc80ce4 -- Fri Jan  1 22:22:00 2010
 atime: 0x4b3dab80:00000000 -- Fri Jan  1 00:00:00 2010
 mtime: 0x4b3dab80:00000000 -- Fri Jan  1 00:00:00 2010
[…]
```

http://blog.commandlinekungfu.com/2010/02/episode-80-time-bandits.html

What's Going on Here?

```
# stat /tmp/testing | tail -3
Access: 2010-01-01 00:00:00.000000000 -0800
Modify: 2010-01-01 00:00:00.000000000 -0800
Change: 2010-03-08 16:42:33.993133369 -0800
# echo 2 >/proc/sys/vm/drop_caches
# stat /tmp/testing | tail -3
Access: 2010-01-01 00:00:00.000000000 -0800
Modify: 2010-01-01 00:00:00.000000000 -0800
Change: 2010-01-01 22:22:00.993133369 -0800
```

- Flushing caches also useful when doing perf analysis

http://blog.commandlinekungfu.com/2010/02/episode-80-time-bandits.html

Stumper Question #1

- Searching a directory structure
- Want to find files containing a particular string
- Only want to look in ASCII text files

```
find /usr/lib -type f | \
   xargs file | egrep -i 'script|ascii|text' | \
   sed 's/:.*//' | xargs grep mystring
```

http://blog.commandlinekungfu.com/2010/02/episode-81-from-mailbag.html

Stumper Question #2

- Two directories
- Each will have some files in common and some unique
- Matching files may have different names between dirs
- Create a list of unique files from both dirs

```
find dir1 dir2 -type f | xargs md5sum | \
   sort -u -k 1,1 | awk '{$1=""; print}'
```

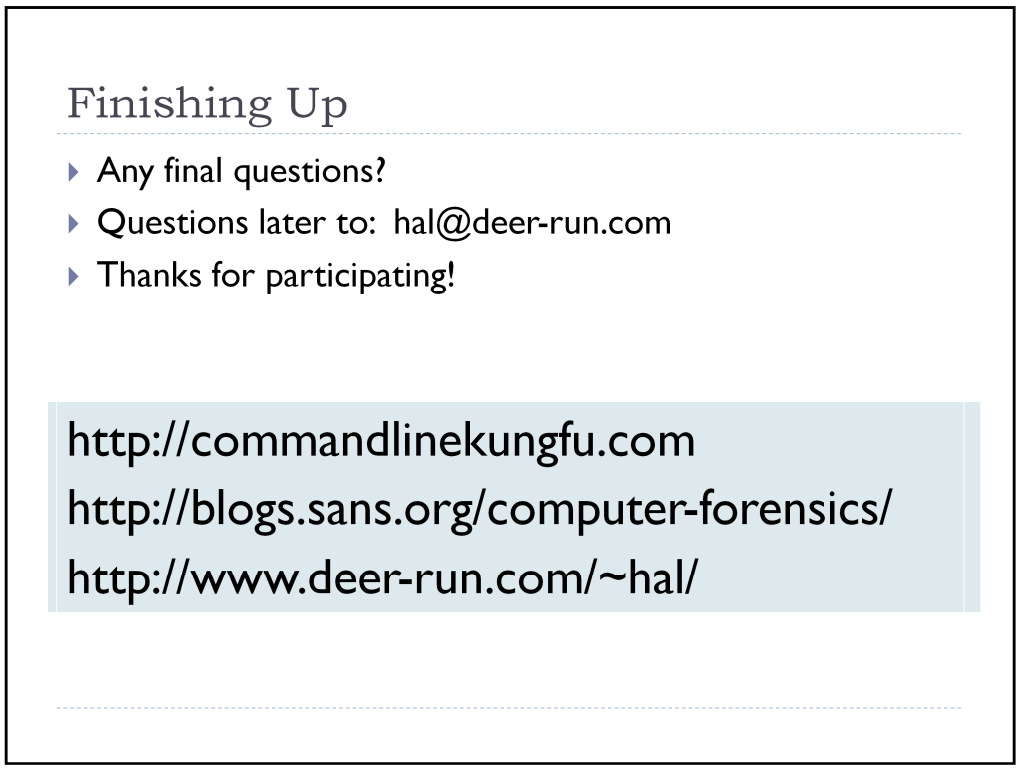
http://blog.commandlinekungfu.com/2010/03/episode-87-making-hash-of-things.html

Stumper Question #3

- List processes listening on network ports
- Output: PID, proc owner, IP, port, and *program binary path*

```
lsof -nPi | awk '/LISTEN/ {print $2, $3, $7, $8}' | \
   sed -r 's/:([0-9]+)$/ \1/' | \
   while read pid rest; do
      echo "$rest" `lsof -a -p $pid -d txt | \
                    awk '/txt/ {print $9, $10}'`;
   done
```

http://blog.commandlinekungfu.com/2010/05/episode-93-of-ports-and-paths.html



Thank you for your time and attention. If you have any questions about the material in this presentation, here's my contact info again:

Hal Pomeranz
Deer Run Associates
PO Box 50638
Eugene, OR 97405

hal@deer-run.com
(541)683-8680
(541)683-8681 (fax)
http://www.deer-run.com/