bash: Secure (bash) Shell Scripts

PLUG North
2009-03-09

PLUG West
2009-03-16

JP Vossen
bashcookbook.com
Secure? Really?

- How can shell scripts be secure when you can read the source code?
- Security by obscurity
  - OpenSSH vs. Windows
- Do what they should
- Not what they shouldn't
- Robust and fail gracefully
- Sanitize input
- Documented
Shebang! [14.2]

- If `#!` Kernel looks for interpreter
- Linux kernel accepts a single argument; BSD & Solaris accept more

- `#!/bin/bash` -
  - A bit more secure, a bit less portable

- `#!/usr/bin/env bash`
  - Uses `$PATH`!

- Details:
  [Details](http://www.faqs.org/faqs/unix-faq/faq/part4/section-7.html)
$PATH [14.3,10 15.2]

- Hard-code at top of script
  - Helps for cron

- Don't add the current dir '.' to $PATH
  - But why?
  - DOS/Windows

- POSIX $PATH
  - $PATH=$(getconf PATH)
  - $PATH=`getconf PATH`
  - export?
Aliases [14.4]

- \ prefix prohibits alias expansion
  - Why?

- Examples
  - \cd
  - \unalias -a

- 'help unalias'
Limits [14.6]

- **ulimit** -H -c 0 –
  - -H = hard limit
  - -c 0 = no core dumps
  - – = end of options

- Why?
- 'help ulimit'
Command Hash [14.5]

- hash -r
  - -r = “reset” or clear

- Why?
- 'help hash'
$IFS [14.7]

- $IFS="' \t\n'
  - Not portable; bash and ksh93 only
  - $*, ${!@}, ${!*}, parameter expansion, and programmable completion all use the **first** character of $IFS.

**Breakable:**
- $IFS='○→¶
   
**Unexpected:**
- $IFS='¶
   ○→'
umask [14.8]

- Sets default file and directory creation permissions

- This is a **mask**!
  - 002 = 0774
  - 077 = 0700
  - etc.

- umask 002
“temp” dirs [14.11]

- Yuck:
  - mktemp
  - /dev/urandom

- $RANDOM is great. But not in dash! :-(
- `temp_dir="prefix$RANDOM$RANDOM$RANDOM"
- Why make a “temp” directory?
  - `mkdir -m 0700`

- Examples tarball: `make_temp`
Cleaning up [14.11,23 17.7]

- trap
  - trap "rm -rf $temp_dir" ABRT EXIT HUP INT QUIT

- ~/.bash_logout
  - clear

- Disconnecting inactive sessions
  - SSH: “IdleTimeOut”, but not in OpenSSH
  - Bash: $TMOUT <seconds>
  - X screen saver
World-writeable Dirs [14.9]

- Examples tarball: chkpath.2

```bash
exit_code=0
for dir in ${PATH//:/ }; do
    [ -L "$dir" ] && printf "%b" "symlink, "
    if [ ! -d "$dir" ]; then
        printf "%b" "missing				"
        (( exit_code++ ))
    else
        stat=$(ls -lHd $dir | awk '{print $1, $3, $4}')
        if [ "$($stat | grep '^d.......w. ')" ]; then
            printf "%b" "world writable	$stat 
            (( exit_code++ ))
        else
            printf "%b" "ok		$stat 
        fi
    fi
    printf "%b" "$dir
done
exit $exit_code
```
Chkpath.2 Output

# ./chkpath ; echo $?  
ok  drwxr-xr-x root root  /usr/local/sbin  
ok  drwxr-xr-x root root  /usr/local/bin  
ok  drwxr-xr-x root root  /sbin  
ok  drwxr-xr-x root root  /bin  
ok  drwxr-xr-x root root  /usr/sbin  
ok  drwxr-xr-x root root  /usr/bin  
ok  drwx------ root root  /root/bin  
missing  /does_not_exist  
world writable  drwxrwxrwt root root  /tmp  
symlink, ok  drwxr-xr-x root root  /root/sbin  
2
Validating Input [14.12]

- Why?
  - Hint: e.g. SQL injection (http://xkcd.com/327/)
- Can be tricky
  - Examples tarball: validate_using_case
  - http://www.bashcookbook.com/bashinfo/source/bash-4.0/examples/scripts/shprompt
Setting Permissions [14.13]

- chmod 0755 some_script
- chmod +x some_script

- $ find some_directory -type f -print0 \
  | xargs -0 chmod 0644  # File perms

- $ find some_directory -type d -print0 \
  | xargs -0 chmod 0755  # Dir. perms
Passwords [14.14,20]

• $ ./cheesy_app -u user -p password
  – ps auwx (on Linux, man ps for other OSs)
  – .bash_history
    • Leading space or $HISTIGNORE
  – -c <config file>

• Using in scripts
  – sudo with NOPASSWD
  – Maybe SSH
  – ~/.hidden/
setuid and setgid [14.15]

- OK for directories, since it means something different.
- Linux won't even allow setuid on shell scripts (portability problem).
  - You could do a C wrapper. Don't.
- Use sudo instead
Users [14.16,18,19,22]

- **Non-root**
  - Duh! Ubuntu does this well
- **sudo**
  - Also duh, but... man sudoers
  - Does **lots** more than you think!
- **rbash** Restricted shell in `/etc/passwd`
  - No: cd, redirection, commands/source with /, exec, functions, set +r
  - Can't: change environment variables or built-ins
- **SSH**
  - `~/.ssh/authorized_keys` & “forced commands”
  - command=`"/bin/echo Command was: $SSH_ORIGINAL_COMMAND"`
chroot [14.17]

• Mainframe VMs (since late 1960's!)

Current VMs
  – VMware, VirtualBox, Xen, QEMU, etc.
• BSD jails
• Solaris 10+ “containers” or “zones”
  – http://www.opensolaris.org/os/community/zones/faq/
• Linux chroot
  – Tricky, lame
Questions?

- http://examples.oreilly.com/bashckbk/
- [N.N] = Recipe numbers
- bashcookbook.com
- PLUG Mailing list
- jp@jpsdomain.org