Openwall GNU/*/Linux

Solar Designer
<solar@owl.openwall.com>

Rafal Wojtczuk
<nergal@owl.openwall.com>

Why another Linux distro?

- Aren't major Linux distributions secure?
  - Most care to patch **known** security vulnerabilities which are "**bad enough**", yet do little to prevent vulnerable software from getting into the distribution in the first place
  - There're usually more than just a few pieces of software in a distribution which provide a certain bit of functionality, thereby unnecessarily increasing the risk
    - The number of vulnerabilities affecting each major distribution that hit Bugtraq is high, and those are only the ones which are "bad enough"

Why another Linux distro? (cont.)

- Isn't there already a secure Linux distribution?
  - Most choose software based on security track record
    - A good security track record is no replacement for source code review; unless the software component is very popular, the track record hardly says anything on its **design** and **code quality**
    - It isn't just the **choice** of software which matters
  - There's often an emphasis on kernel modifications
  - It's not the security-related bells and whistles which make a system secure

Openwall GNU/*/Linux (Owl)

A security-enhanced server platform based on

- The Linux kernel and its corresponding utilities
- GNU software
- Many BSD-derived components, including those ported to Linux specifically for use in Owl
- Other free software from various authors
- Free software developed by Openwall team members, including specifically for Owl

Owl: Features

- A base for installing whatever software is generally available for GNU/*/Linux systems (including commercial and closed-source)
- Includes a growing set of integrated Internet services
- Includes a complete build environment ("make buildworld")
- Supports multiple architectures (currently x86, SPARC, Alpha)

Owl: Approach to security

- Software design and code quality are first priority
- Source code review
  - Pieces of code which are typically run with privileges greater than those of a regular user and/or typically process data obtained over a network are audited before the corresponding software component is included; this applies to
    - relevant code paths in many of the system libraries
    - all SUID/SGID programs
    - all daemons and network services

Owl: Approach to security (cont.)

- Software modifications in order to
  - apply the least privilege principle
  - introduce privilege separation
- Safe default configuration
- As the project evolves, many of the software components will be replaced with ones of our own

Owl: Approach to security (cont.)

- Policy enforcement and integrity checking
- "Strong" cryptography within core OS components
- "Hardening" to reduce likelihood and/or impact of successful real-world attacks on insecure third-party software one might install on the system
- A wide range of security tools available for use "out of the box"

Owl: Build environment

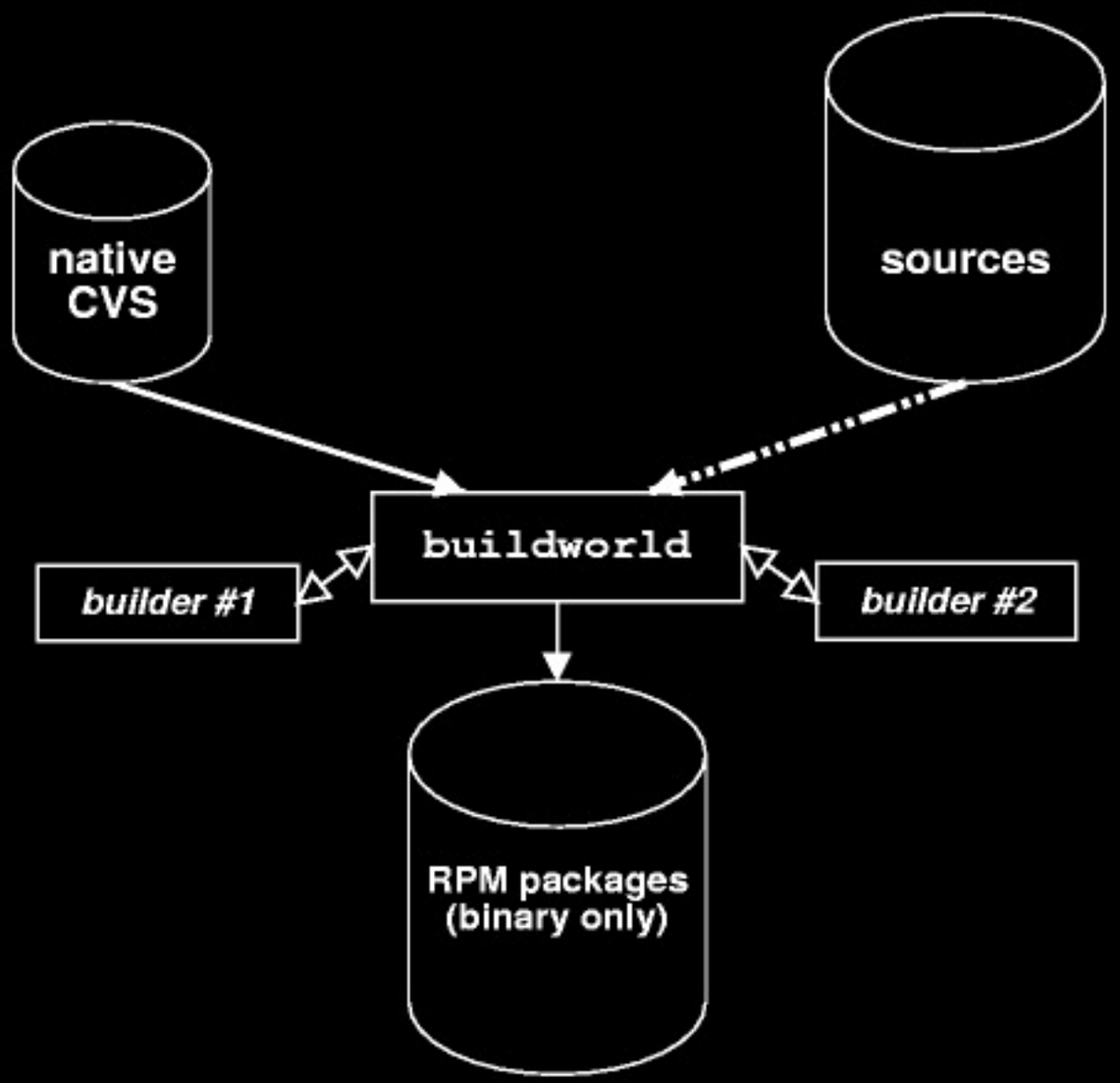
The Owl userland is maintained similarly to *BSD ports/packages and may be rebuilt with one simple command ("make buildworld")

Some build times:

Dual Pentium III, 1.266 GHz, 2 GB	0:30
UltraSparc IIi, 400 MHz, 256 MB	3:50
Alpha 21164PC, 533 MHz, 128 MB	5:30

(Yes, gcc is this slow on Alpha)

The build times will increase as we add more packages and update to new versions of software already in Owl



Owl: Developed software

- Portable
  - pam_mktemp, pam_passwdqc, pam_userpass; popa3d; scanlogd; libnids; John the Ripper
- Semi-portable
  - crypt_blowfish, tcb (libtcb, libnss_tcb, pam_tcb)
- Owl-specific
  - owl-control
    - intended for Owl only, but now also used by ALT Linux
  - Startup scripts, the build environment, and so on

Owl: Ported software

- Several software components have been ported from OpenBSD (with our usual source code review and modifications)
  - mtree
    - and we actually build the initial filesystem hierarchy with mtree
  - Vixie Cron (crontab, crond)
    - with modifications for SGID crontab(1)
  - telnet, telnetd
    - with modifications to introduce privilege separation
  - netcat (nc)
  - mailx

Owl: Modified software

- Essentially all of it
  - on average 4 patch files per package
  - (the most important) half of the patches originate in Owl
  - the other half has been contributed or imported from various other distributions (including *BSD's)
    - with appropriate credit given in each patch file name

```
owl!build:~/native/Owl/packages/tcp_wrappers$ wc -c *.diff
  22005 tcp_wrappers_7.6-openbsd-owl-cleanups.diff
   4272 tcp_wrappers_7.6-openbsd-owl-ip-options.diff
   4088 tcp_wrappers_7.6-owl-Makefile.diff
   1866 tcp_wrappers_7.6-owl-safe_finger.diff
   4938 tcp_wrappers_7.6-steveg-owl-match.diff
```

Owl: crontab / crond

- What privileges does crontab(1) require?
  - Ability to insert jobs into crond(8) spool
- The least privilege principle in the flesh

```
owl!root:/var/spool/cron# ls -ld . joe
drwx-wx--T   root    crontab         1024 Nov  5 14:10 .
-rw-------   joe     crontab          493 Apr  3  2001 joe

owl!root:/usr/bin# ls -l crontab
-rwx--s--x   root    crontab       21116 Nov  5 14:10 crontab
```

  - crond(8) must not blindly trust its spool directory (and ours doesn't)

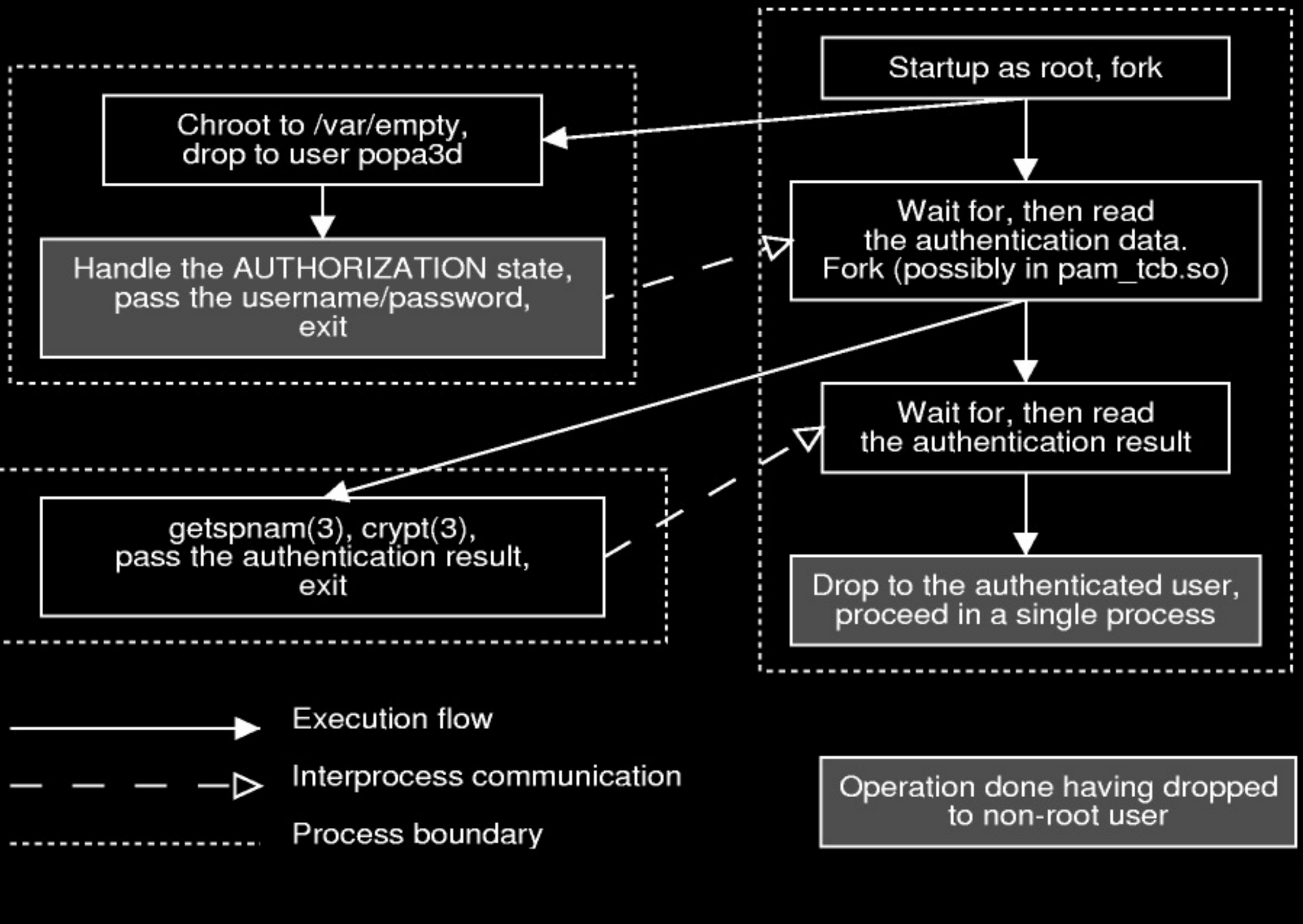
Owl: syslogd architecture

- Initialization as root
  - Bind a socket to **/dev/log**
  - Process **/etc/syslog.conf**, open appropriate log files
  - Drop to user/group **syslogd**
- Normal operation as user **syslogd**
  - Read from **/dev/log**, write to the log files
- In order to be able to reopen the log files on SIGHUP, they must be made writable to user or group **syslogd** when rotated

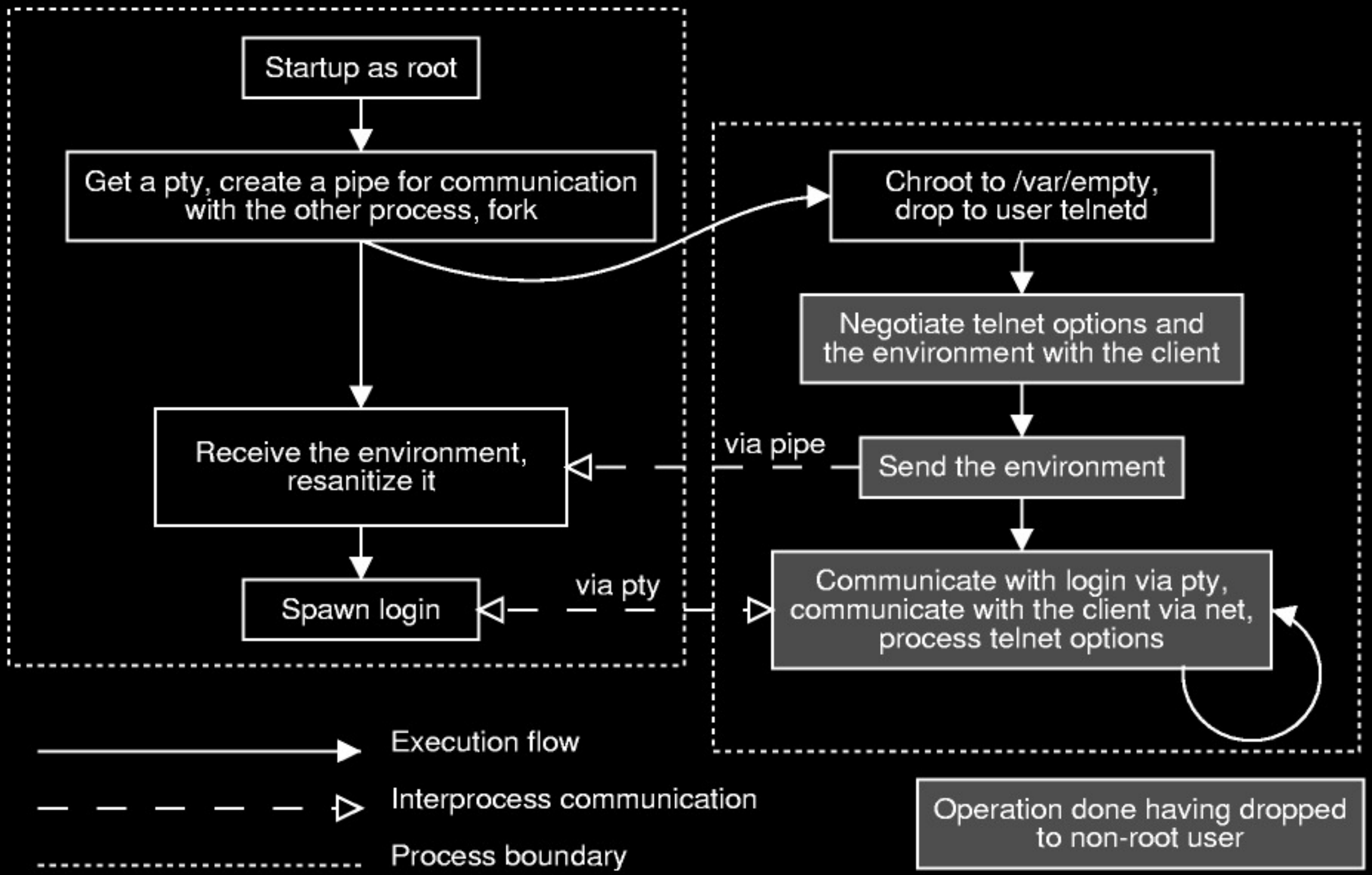
Owl: klogd architecture

- Initialization as root
  - Open **/proc/kmsg** and **/dev/log**, retain the open fd's
  - Open **/dev/kmem** and **System.map**, read relevant data, close them
  - Chroot to **/var/empty**
  - Drop to user **klogd**
- Normal operation as user **klogd**, in the chrooted environment
  - Read from the **/proc/kmsg** fd, format the message, and write it to the **/dev/log** fd

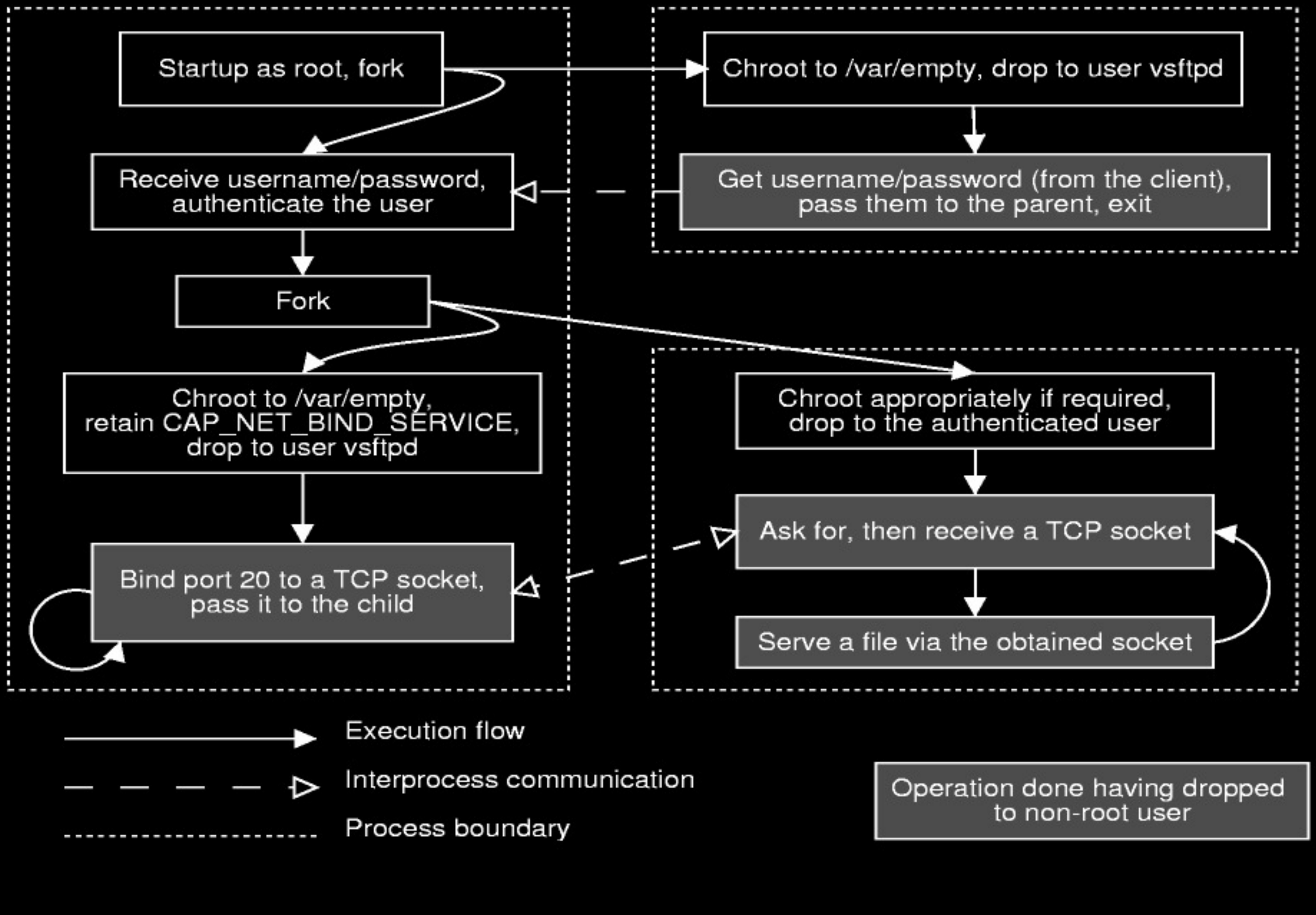
Owl: popa3d architecture



Owl: telnetd architecture



Owl: vsftpd architecture



Traditional password shadowing

- Password hashes and aging information of all users are stored in a single file
  - passwd(1) possesses the privilege to alter **all** entries in the shadow file
    - The traditional filesystem layout forces passwd(1) to be SUID root
  - chage(1) possesses the privilege to read **all** entries in the shadow file
- A passwd process compromise is fatal
- The problem cannot be fixed by assigning a dedicated user for **/etc/shadow** accesses

Owl: tcb - the alternative to shadow

- Each user is assigned a separate shadow file
- Each user is the owner of their shadow file
- Access to shadow files is group-restricted to allow for password policy enforcement
- The move to tcb is transparent for existing applications which rely on interfaces such as getspnam(3) (and thus on NSS) or PAM; no modifications to application sources are needed

Owl: tcb: Filesystem layout

```
owl!root:~# ls -ld /etc/tcb/
drwx--x---   root    shadow  1024 Nov 27 12:14 /etc/tcb/

owl!root:~# ls -l /etc/tcb/
drwx--s---   root    auth    1024 Nov 27 12:14 root
drwx--s---   joe     auth    1024 Nov 27 12:14 joe

owl!root:~# ls -l /etc/tcb/joe/
-rw-r-----   joe     auth      84 Nov 27 12:14 shadow

owl!root:~# cat /etc/tcb/joe/shadow
joe:$2a$08$ghnh1Q5K6kE24bY9xqQa5uSXwG2YO4O5lbj.yfLKp8BVFBusq
Lwxi:11320:0:99999:7:::
```

- The per-user directories are also used as scratch space for temporary and lock files which are needed during password change

Owl: tcb: Required privileges

- passwd(1) is made SGID **shadow**
- chage(1) is SGID **shadow**
  - A possible compromise would only let one bypass password policy enforcement for their own account
- Group **auth** may be used to grant a process read access to all password hashes should the need arise
- No real need for any SUID binaries on the entire system

Owl: tcb: Components

- libtcb, the auxiliary library used by almost all of the tcb suite
  - Provides functions for locking and accessing tcb shadow files safely
- libnss_tcb, the NSS module
  - Provides getspnam(3) and related functions
  - When running as root, the **/etc/tcb/*/shadow** files are accessed with the proper effective credentials and treated as untrusted input

Owl: tcb: Components (cont.)

- pam_tcb, the PAM module
  - Provides functionality for all four PAM management groups
  - Supports **/etc/passwd**, **/etc/shadow**, **/etc/tcb/** directory structure, NIS, and NIS+ for password changes
  - Supports arbitrary password hashing methods
  - Optional forking to keep address space clean
  - Backwards compatible with Linux-PAM pam_unix and pam_pwdb but offers additional functionality and better code quality

Owl: tcb: Components (cont.)

- tcb_convert and tcb_unconvert
  - Easy conversion between **/etc/tcb/*** and traditional **/etc/shadow** databases
- The shadow suite utilities
  - Non-trivial patching has been applied to the sources of most shadow suite utilities
  - The invocation syntax remained unchanged
  - A setting in **/etc/login.defs** specifies whether the utilities should adhere to the tcb scheme

Owl: Further information

- The Openwall GNU/*/Linux homepage is

  http://www.openwall.com/Owl/

- Any questions?