An Overview of Security in the FreeBSD Kernel

Brought to you by

Dr. Marshall Kirk McKusick

2013 BSDCan Conference
May 17, 2013

University of Ottawa
Ottawa, Canada

Security Mindset

Security is part of the design, not added later

From its beginning UNIX identified users and used those identities

- access control to files
- manipulation control of processes
- access control to devices
- limited privilege expansion using *setuid* ( ) and *setgid* ( )

Over time these basic controls have been refined though still remain intact more than 40 years later

Trusted Computing Base

Set of things that have to be secure for system to be secure

- Kernel
- Boot scripts
- Core utilities (shell, login, ifconfig, etc)
- Libraries used by core utilities

Solid crypto support

- OpenSSH
- OpenSSL
- IPSEC
- GBDE
- GELI
- Hardware crypto

Overview

Immutable and Append-only Flags

- Tamperproof critical files and logs

Jails

- Lightweight FreeBSD virtual machine

Access control lists (ACL)

- Discretionary access control to files and directories

Mandatory access control (MAC)

- Systemwide controlled information flow between files and programs

Privilege

- Subdivision of root privileges

Auditing

- Accountability and intrusion detection

Capsicum

- Sandboxing of process rights

Immutable and Append-only Flags

- Immutable file may not be changed, moved, or deleted
- Append-only file is immutable except that it may be appended
- User append-only and immutable flags may be toggled by owner or root
- Root append-only and immutable flags may not be cleared when system is secure

- System secure levels:
  - -1 always insecure (must be compiled into kernel)
  - 0 insecure mode (normally single user)
  - 1 secure mode (normally multiuser)
  - 2 very secure mode (at system admin discretion)

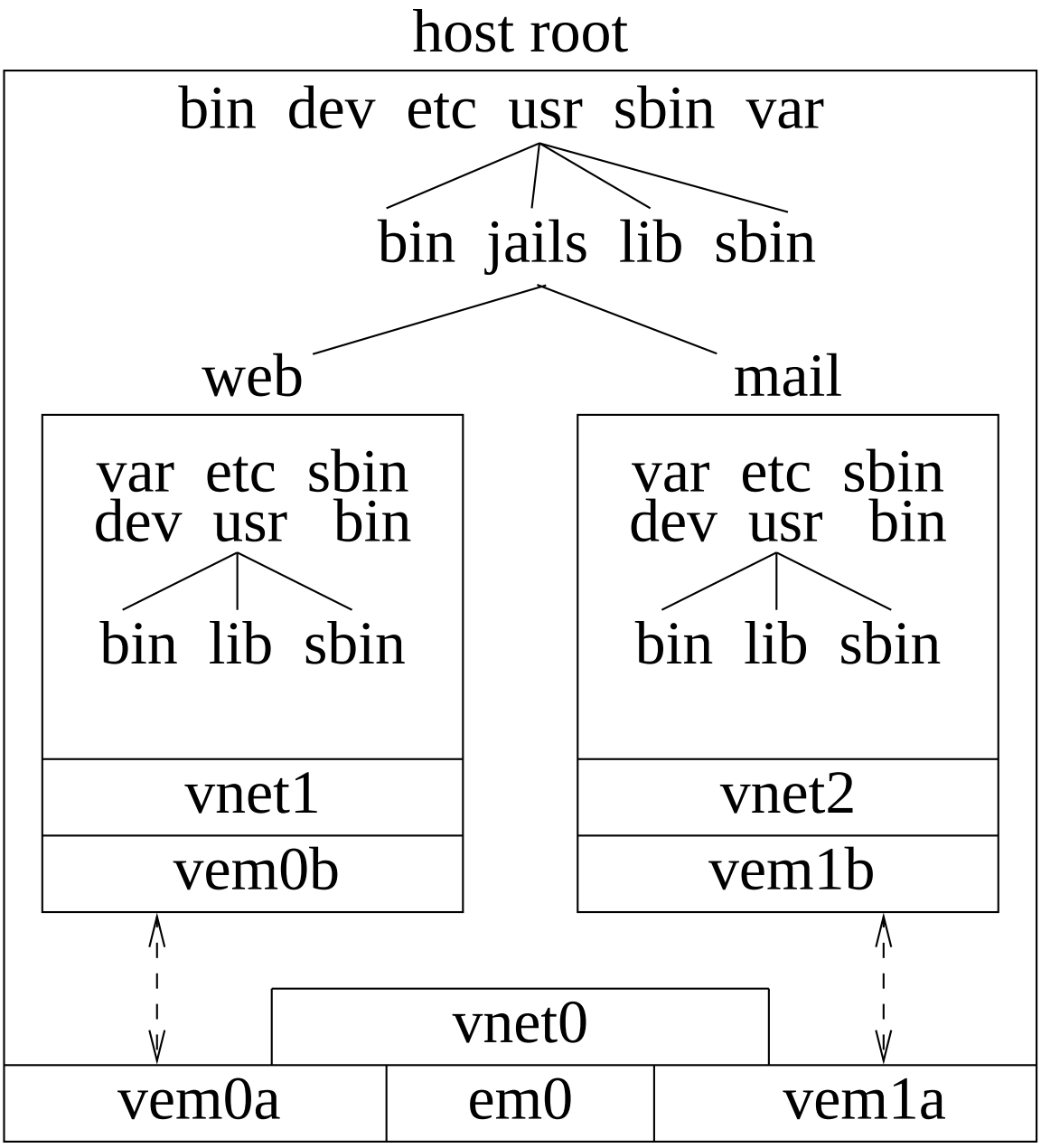
- Secure mode prevents writing /dev/kmem, /dev/mem, and mounted disks
- Very secure mode additionally prevents writing any disk or rebooting

Immutable Limitations

- Immutable files can only be updated when system is single-user
- Append-only files can only be rotated when system is single-user
- Direct hardware access is restricted
- All startup activities must be protected
  - Startup scripts and their containing directories
  - All binaries executed during startup
  - All libraries used during startup
  - Many configuration files used during startup

Jails

Create a group of processes with their own root-administered environment



Jail Rules

Permitted

- running or signalling processes within jail
- changes to files within jail
- binding ports to jail’s IP addresses
- accessing raw, divert, or routing sockets on jail’s virtual network interfaces

Not permitted

- getting information on processes outside of the jail
- changing kernel variables
- mounting or unmounting filesystems
- modifying physical network interfaces or configurations
- rebooting

Access Control Lists

File permission bits

- file permission bits are three entries in the ACL itself
- permits full backward compatibility with historical implementations

ACL capabilities:

- read, write, execute, lookup, and admin permissions
- list of users each with own permissions
- list of groups each with own permissions
- permissions for all others

Default/inheritable ACL's that propagate down the file hierarchy

Two user-level commands:

- *getfacl* - get file ACL permissions
- *setfacl* - set file ACL permissions

Access Control List Semantics

Support for POSIX.1e and NFSv4 semantics

- By design, NFSv4 semantics are very similar to Windows filesystem ACL semantics
- UFS implements both POSIX.1e and NFSv4 semantics (specified at boot time)
- ZFS implements only NFSv4 semantics
- NFSv4 uses inheritable ACLs rather than the default ACL in POSIX.1e
- FreeBSD uses the same command-line tools and APIs for both ACL types

Privilege

Each superuser privilege is identified and treated separately

Nearly 200 defined in **/sys/sys/priv.h**, some examples:

- PRIV_ACCT − Manage process accounting.
- PRIV_MAXPROC − Exceed system processes limit.
- PRIV_SETDUMPER − Configure dump device.
- PRIV_REBOOT − Can reboot system.
- PRIV_SWAPON − Add swap space.
- PRIV_MSGBUF − Read kernel message buffer.
- PRIV_KLD_LOAD − Load a kernel module.
- PRIV_ADJTIME − Set time adjustment.
- PRIV_SETTIMEOFDAY − Can set time of day.
- PRIV_VFS_WRITE − Override vnode write permission.

Priviledge Applied

Privilege checks cover all areas of the system

- network configuration and filtering
- filesystem mounting, unmounting, and exporting
- accessing or modifying kernel data and modules
- many others

Each privilege has three properties applied to a process or a file

- permitted: whether the process or file may ever have the privilege
- inheritable: whether the process or file may grant the privilege
- effective: whether the process or file can currently use the privilege

Access to privilege is done with MAC modules via the *priv_check* ( ) function.

Mandatory Access Control

Allows arbitrary security policies to be added to the system using labels and an expansion of traditional root access controls

Controls access/use of:

- files, pipes, and sockets
- kernel load-modules
- network interface configuration
- packet filtering
- process execution, visibility, signalling, and tracing
- file mapping
- kernel data
- accounting information
- NFS exports
- swapping

Auditing

Accountability and intrusion detection

Based on Open Basic Security Module (OpenBSM)

Generate records for kernel events involving

- access control
- authentication
- security management
- audit management
- user-level audit reports

Volume of audit trail is controllable

- audit preselection policy
- **auditreduce** to thin audit logs

User credentials can be augmented with an audit identifier (AUID)

- Holds terminal and session to be added to each audit record
- audit mask to subset global audit preselection policy

Audit Handling

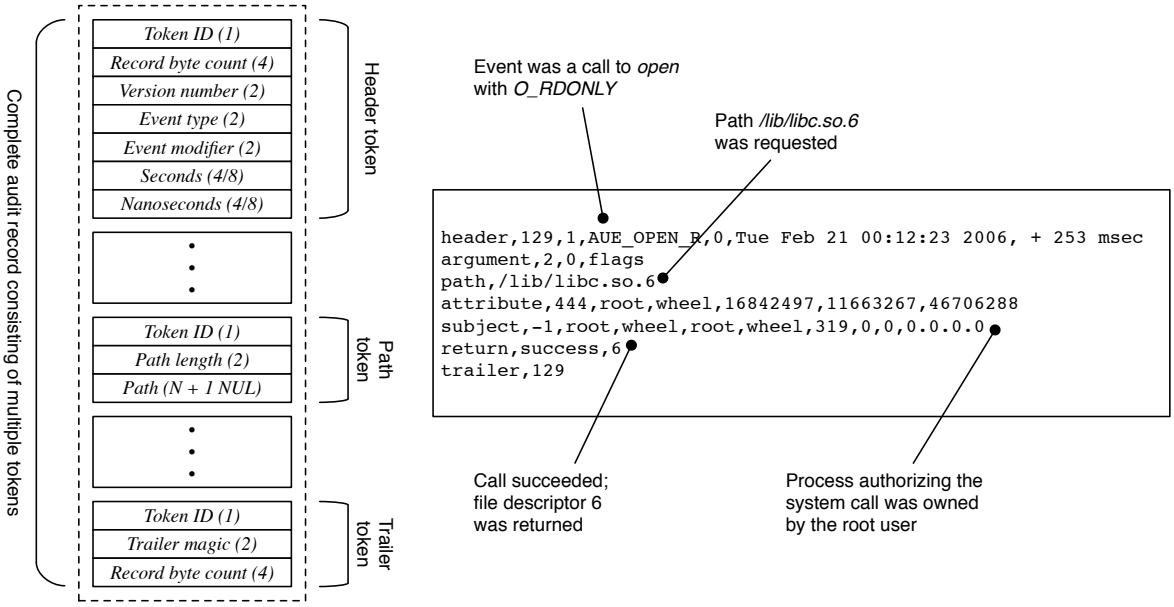
auditd daemon

- manages data collection
- content selection including selection of records collected
- responds to events such as running low on disk space

auditd daemon starts a kernel thread that manages record distribution

- stored in local filesystem
- sent elsewhere for storage
- sent to intrusion detection daemon

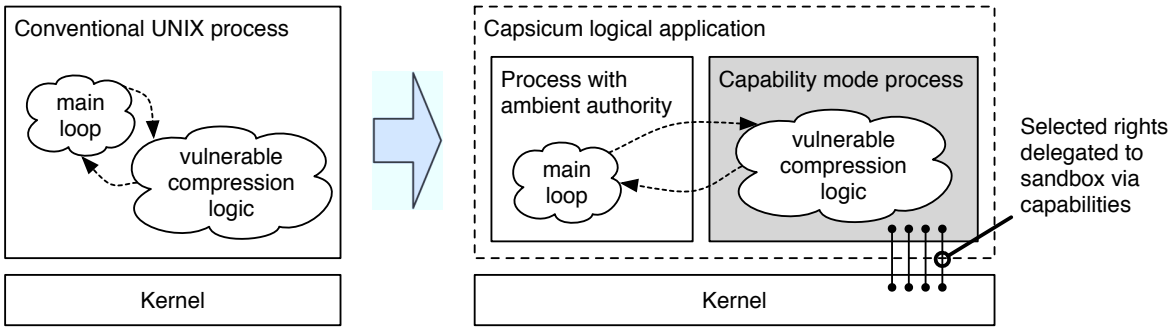
Example audit record



Capsicum

Sandboxing of limited trust modules

- A small process with full privileges
- Untrusted libraries/modules run in separate process with access limited to minimal set of things that they need



Using Capsicum

- Process put into capability mode with *cap_enter* ( )
- Once in capability mode, cannot exit
- Can only work with its own file descriptors
- No access to filesystem namespace (e.g., *open* ( ) will fail but *openat* ( ) will work if given a descriptor open on a directory from which to start.

Sample Capsicum Capabilities

A set of rights is delegated to each descriptor

Sixty defined in **/sys/sys/capability.h**, some examples:

- CAP_READ − Read or receive
- CAP_WRITE − Write or send
- CAP_SEEK − Modify file descriptor offset
- CAP_FCHFLAGS − Set file flags
- CAP_FCHDIR − Set working directory
- CAP_FCHMOD − Change file mode
- CAP_FCHOWN − Change file owner
- CAP_LOOKUP − Use as starting directory for at operations
- CAP_POLL_EVENT − Test for events using select, poll, kqueue
- CAP_POST_EVENT − Post an event to kqueue
- CAP_ACCEPT − Accept sockets
- CAP_LISTEN − Set up a listen socket

Questions

Marshall Kirk McKusick
<mckusick@mckusick.com>
http://www.mckusick.com