Nuts and Bolts of Enhanced Security Management for Tru64 UNIX

Martin Moore

Team Leader, Tru64 UNIX Support
Hewlett-Packard

Session topics

- Topics
  - Overview – what is Enhanced Security?
  - Enhanced Security architecture
  - Configuring Enhanced Security features
  - Additional Tru64 UNIX security features
  - Q & A
- Not a topic
  - Hardening system for maximum security
    - See session 3760, “Securing Tru64 UNIX”

What is Enhanced security?

- Optional OS subsets (OSFC2SECnnn and OSFXC2SECnnn) that provide additional, configurable security features
- No extra license needed
- Can be configured to reach the C2 class of trust defined by TCSEC (US)
- Also meets the F-C2 functional class defined by ITSEC (EU)

Security classes (lowest to highest)

TCSEC (US)	ITSEC (EU)	TCSEC definition
D	-	Minimal security
C1	E1, F-C1	Discretionary security protection
C2	E2, F-C2	Controlled access protection
B1	E3, F-B1	Labeled protection
B2	E4, F-B2	Structured protection
B3	E5, F-B3	Security domains
A1	E6, F-B3	Verified design

Enhanced security architecture

- Security Integration Architecture (SIA)
- Enhanced security daemon (prpasswdd)
- Authentication database

Security Integration Architecture (SIA)

- O/S layer that provides interface to code that depends on security mechanisms: user authentication, password changes, etc.
- Controlled by */etc/sia/matrix.conf* – defines which libraries to use for security-dependent calls (e.g. changing password)
- Changing security level changes matrix.conf
  - Some layered products (DCE, ASU, etc.) also modify the file
- SIA log file *(/var/adm/sialog*) records SIA activity (e.g., su's) – touch the file to start logging
- Can be used to customize security-sensitive commands
  - See Security manual chapter 20 for much more info

Sample matrix.conf file

```
siad_setgrent=(BSD,libc.so)
siad_endgrent=(BSD,libc.so)
siad_getgrent=(BSD,libc.so)
siad_getgrnam=(BSD,libc.so)
siad_getgrgid=(BSD,libc.so)
   :
siad_init=(OSFC2,/usr/shlib/libsecurity.so)
siad_chg_finger=(OSFC2,/usr/shlib/libsecurity.so
  )
siad_chg_password=(OSFC2,/usr/shlib/libsecurity.
  so)
siad_chg_shell=(OSFC2,/usr/shlib/libsecurity.so)
```

…etc.

Enhanced security daemon

- */usr/sbin/prpasswdd*
- Introduced in V5 to handle writes to security databases, avoiding file lock contention
- Two instances (parent and child) should be running at all times
- If you're having unexplained login problems in V5 Enhanced security, try restarting prpasswdd:

```
# /sbin/init.d/prpasswd restart
```

(Note: two "d"s in daemon name, only one in script name.)

Authentication database

- A set of five component databases that contain all Enhanced Security information (man page: *authcap*(4))
  - Protected password
  - Terminal control
  - System default
  - Device assignment
  - File control
- Use *edauth*(8) to manipulate databases
- Other useful commands:
  - *authck*(8): check database consistency
  - *convauth*(8): convert old (pre-V4) database to new
  - *convuser*(8): convert profile from Base to Enhanced

Authentication database components

- Protected password database – *prpasswd*(4)
  - User profile and password information
  - */tcb/files/auth.db* (UID 0-99)
  - */var/tcb/files/auth.db* (UID 100+)
  - Fields begin with “u_”
- Terminal control database – ttys(4)
  - Terminal login control profile
  - */etc/auth/system/ttys.db*
  - Fields begin with “t_”

Auth database components (2)

- System default database – default(4)
  - System-wide security defaults
  - Default values for fields in other components
  - */etc/auth/system/default*
  - System-wide default fields begin with "d_"
  - "t_", "u_", and "v_" fields may also appear
- Device assignment database – devassign(4)
  - Login control for terminals and X devices
  - */etc/auth/system/devassign*
  - Fields begin with "v_"

Auth database components (3)

- File control database – files(4)
  - Maintains system file integrity
  - */etc/auth/system/files*
  - Fields begin with “f_”
- Each component database is a set of entries (e.g., prpasswd contains one entry for each user)
- Each entry consists of the entry name, one or more data fields, and the end-of-entry field “chkent”

Authentication database format

- Fields are colon-separated, and one of 3 types:
  - Integer: `<name>#<value>` `u_id#115`
  - String: `<name>=<value>` `u_name=martin`
  - Boolean: `<name>` (if true) `u_lock`
    `<name>@` (if false) `u_lock@`

- Example:

```
# edauth -g -dt console
console:\
  :t_devname=console:t_uid=root:\
  :t_logtime#1053073399:chkent:
```

(Last login on console was by root at time 1053073399.)

Authentication database format (2)

- Some integer values are time values
  - Duration in seconds (1 day = 86400 sec)
  - Absolute times (seconds since start of 1970)
    - "1053073399" (previous slide) = 08:23:19 GMT, 16 May 03
    - Time value of zero means never or infinity
  - Kdbx macro "ctime" translates into meaningful format:

```
# echo "ctime 1053073399" | kdbx -k /vmunix \ |
  tail -1
<cr><cr>
Fri May 16 03:23:19 EST 2003
```

Migrating Users from V4 to V5

- The internal format of auth.db changed in V5
  - Can’t just copy over V4 auth.db files!
- On the V4 system:

```
# edauth -g > auth.db.text

# tar cvf auth.db.tar auth.db.text /etc/group
/etc/passwd
```

- Copy auth.db.tar to the V5 system and do the following:

```
#cd /etc

#cp -p group group.last

#cp -p passwd passwd.last

#tar xpf auth.db.tar

#edauth -s < auth.db.text
```

Enhanced security features

- Distributed passwords (aka shadow passwords)
- Login controls
- Password controls
- Account templates

Shadow passwords

- By default, encrypted passwords are visible in world-readable */etc/passwd*; open to crack attack
- Shadow passwords are encrypted passwords in a non-visible location (prpasswd database)
- V5 provides ability to easily select shadow passwords without other C2 features

Login controls

- Recording of last terminal and time of last successful login and last login failure
- Account disabled after too many consecutive failures; limit configurable on a per-user basis
- Similar lockout configurable per terminal
- Minimum time between login attempts
- Maximum time for login attempt to complete
- Day/time login restrictions for individual users
- Account lifetime (account retired when reached)

Password controls

- Maximum (up to 80) and minimum password length
- Password expiration time (if not changed in this time, must be changed at next login)
- Password lifetime (if not changed in this time, account disabled)
- System-generated passwords (several flavors) or user-chosen passwords, configurable per user
- Password history to prevent re-use (depth configurable per user)

Password controls (2)

- Optional triviality checks – built-in & site-specified
- Built-in: see *acceptable_password*(3) man page
  - No palindromes, login or group names, or dictionary words as defined by *spell(1)*
  - Controlled by u_restrict (Boolean)
- Site-specified: see */tcb/bin/pwpolicy* comments
  - pwpolicy is template/placeholder
  - Specify your own callout script/exe with secconfig
  - Specified by u_policy (string – filename)
  - Powerful hook into security authentication
    - http://users.rcn.com/spiderb/sec/site-pwpolicy.c.txt

Locked, disabled, and retired accounts

- Locked – administratively locked by superuser
- Retired – account terminated, never to be reused
- Locked and retired accounts are both disabled; “disabled” simply means that user can’t log in
- Accounts are also disabled by system for violating limits, e.g., too many login failures
- Locked account must be unlocked by superuser
- Retired account can’t be unretired (in strict C2)

Locked, disabled, and retired (2)

- *dxaccounts*(8) indicates these states as follows:
  - Locked: Padlock
  - Retired: Red “No” symbol (circle + diagonal line)
  - Disabled by system: Red circle + white X (V5.1 & up)
- Admin can re-enable disabled accounts with *dxaccounts*, or set a grace period for users to login and remove disabling condition

```
# usermod -x grace_limit=1 <username>
```

  (sets grace period of 1 day for disabled user)

- Or you can remove the disabling condition by editing the user profile with edauth
  - Example: set u_numunsuclog to zero to clear count of unsuccessful login attempts

“Account is disabled” causes

- This message at login can mean any of the following:
  - Administrative lock (“u_lock” present in profile)
  - User on vacation (defined by u_vacation)
  - Password lifetime exceeded
    - Time of last successful password change (u_succhg) is more than <u_life> seconds in the past
  - Account inactive too long
    - Last successful login (u_suclog) is more than <u_max_login_intvl> seconds in the past
  - Too many login failures
    - Number of failures (u_numunsuclog) equals or exceeds maximum number of login attempts (u_maxtries)
    - Automatic reset after <u_unlock> seconds

Configuring Enhanced security

- Ensure subsets are installed
- Run “*sysman secconfig*” (V5) or “*secsetup*” (V4)
- Choose ENHANCED
- Choose Enhanced security profile (V5)
  - SHADOW (Shadow passwords only)
  - UPGRADE (During rolling cluster upgrade only)
  - CUSTOM (Customize enhanced features)
- Reboot needed to switch from Base to Enhanced

Customizing Enhanced security features

- CUSTOM defaults:
  - Login successes and failures are logged
  - Null passwords are not allowed
  - Password expiration = 26 weeks
  - Password lifetime = 52 weeks
- Selecting CUSTOM brings you to Custom Options screen to customize common features
  - Leave "Password Encryption Algorithm" as "BigCrypt"
- Must edit database (*edauth*) for uncommon ones

Customizing system options

- secconfig screen to enable/disable additional security features:
  - Segment sharing
  - Execute bit set only by root
  - Access Control Lists (ACL’s)
- Not technically part of Enhanced Security – independent of security level and of each other
- More on these later

Enhanced security performance

- For the most part, no difference in performance
- Kernel overhead is negligible
- Exception: database updates to record login attempts, especially if numerous and/or frequent logins
- Tradeoff: selectively disable some logging to improve performance (at the expense of security)
  - Logins by terminal (success or failure)
  - Successful logins for user
  - Login failures for each user

Enhanced security and NIS

- Protected password database can be NIS-served
  - Restrictions in mixed-OS NIS environments
- The same NIS domain can include both base and enhanced security clients
- Tricky to set up; see Security manual chapter 9
- Logging of login attempts caused a potential performance bottleneck in V4; also required NIS master to always be up
- In V5, logging can be disabled (see previous slide) to avoid these problems

Enhanced security in clusters

- All members must be at the same security level
- Tricky in V4 (TCR 1.x), particularly before 4.0F
- Much easier in V5
- Strongly recommend configuring Enhanced security on first member before creating cluster
- To upgrade existing cluster, go to UPGRADE and reboot each member in turn
- Then go to SHADOW or CUSTOM and finish configuration

Enhanced security "gotchas"

- 4.0F -> 5.0A update installation has problems with Enhanced security
  - See fix and instructions in 5.0A patch kit
  - Or: go back to Base security during the upgrade
  - Or: avoid this upgrade path if possible
    - Use 4.0F -> 4.0G -> 5.1 -> 5.1B rather than 4.0F -> 5.0A -> 5.1A -> 5.1B
- Authentication database changes are logged in */var/tcb/files/dblogs*
  - If not pruned, could eventually fill up /var
    - Sysman secconfig provides option to schedule a cron job to prune log files
  - See Security manual ch. 6 for security database utilities

Gotchas (2)

- Rolling upgrade of cluster from V5.1A + pk5 or lower, to V5.1B + pk3 or higher
  - prpasswdd inter-node communication changed
  - After roll of first member, all logins will hang (thus, you can't log back in to complete the upgrade!)
  - Workaround: disable prpasswdd before roll phase

```
# cd /sbin/rc3.d
# mv S22prpasswd disable.S22prpasswd
# /sbin/init.d/prpasswd stop   ← on all members
```

  - After rolling upgrade finishes, re-enable prpasswdd

```
# cd /sbin/rc3.d
# mv disable.S22prpasswd S22prpasswd
# /sbin/init.d/prpasswd start   ← on all members
```

Gotchas (3)

- *prpasswdd* uses cluster alias to communicate
  - So all cluster members must be members of the alias
- Base security encrypts only the first 8 characters of password; Enhanced encrypts the entire string
  - In older versions: after switching to Enhanced, log in with only first 8 characters if using a longer password
  - In recent versions, system handles transition correctly via u_oldcrypt and u_newcrypt (don't modify these!)
  - When running sysman secconfig, leave "Password Encryption Algorithm" option set to "BigCrypt"

Administration tools

- Several options exist for day-to-day account management (creating, modifying, locking, etc.):
- Account Manager GUI (*dxaccounts*)
- *Sysman accounts* (GUI or character cell)
  - Similar to dxaccounts, but less powerful
- Command line utilities
  - *useradd, usermod, userdel*
  - *groupadd, groupmod, groupdel*
- Use *edauth* for low-level database manipulation

Other security features

- Segment sharing
- Execute bit protection
- Access Control Lists (ACL's)
- Division of Privilege (DoP)
- Auditing
- Secure Console

Segment sharing

- Page table sharing allows other processes to read text segments (not data) of shared libraries – regardless of their file permissions
- Almost always a non-issue
- Enabled by default; leave it enabled unless you KNOW you need to disable it
- Disabling causes all processes to load private copies of all shared libraries
  - Consumes vast amounts of memory, leading to performance degradation

Execute bit protection

- Feature added in V5 to prevent non-root users from creating executables (e.g., on firewall systems)
- When enabled, non-root users can’t set execute permission bits on any file, even their own
- Disabled by default
- Kernel parameter “noadd_exec_access” in vfs subsystem; 0 = disabled, 1 = enabled
- Reboot needed to change state

Access control lists (ACL’s)

- Increased granularity of access control beyond traditional UNIX user/group/other scheme
- In V5, controlled by parameter “acl_mode” in sec subsystem; can be enabled/disabled dynamically
- *getacl*(1), *setacl*(1) to display/control ACL’s; *dxsetacl*(8X) for graphical interface
- An ACL consists of access control entries for users, groups, and others

Example ACL

```
-rw-r--r--   1 martin   unix   0 Aug 27 10:43 test

# file: test
# owner: martin
# group: unix
#
user::rw-             ← same as "user" bits in ls -l
user:hancock:rw-
user:ellis:---
group::r--            ← same as "group" bits in ls -l
group:staff:--x
other::r--            ← same as "other" bits in ls -l
```

More on ACL’s

- Multiple entry resolution
  - User entry supersedes group entry
  - User in multiple groups gets all their privileges
- Directories have up to 3 different ACL’s
  - Access (controls access to directory)
  - Default access (inherited by new files)
  - Default directory (inherited by new directories)
- ACL’s are stored in property lists; dump and vdump backup and restore these properly

ACL’s and NFS

- Server and client must both have ACL’s enabled
  - As such, won’t work in mixed-vendor configurations
  - Parameter nfs_flatten_mode (in sec subsystem) defines interpretation of ACL’s to NFS V2 clients
- Server must run the property list daemon, *proplistd*(8)
- Client must mount with “proplist” option. An entry in /etc/fstab might look like this:

```
student:/home /nfs_home nfs rw,proplist 0 0
```

Division of Privilege

- Traditional UNIX privileges are all-or-nothing
- *dop*(8) allows you to grant privileges for specific operations to users or groups
- Introduced in V4, but only for system use; not really usable for admins until V5
- Over 20 pre-defined privilege classes, e.g. AccountManagement to add/modify/delete users
- You can define your own privileges (a bit tricky)
- *Sysman dopconfig* for configuration and help

Auditing

- Auditing lets you track system events down to the system call level
- *Sysman auditconfig* to configure
  - Several pre-defined audit profiles, e.g., Desktop, NIS Server, Timesharing, etc.
- *dxaudit*(8X), *auditd*(8), *audit_tool*(8) to manage
- Object selection allows you to focus on specific files
- Audit only what you really want to look at!
  - Too much can drown you in data
  - Does have a performance impact

Per-User Auditing

- Enhanced security has hooks for per-user auditing
  - u_auditmask field in protected password entry
- Useful for tracking login/logout for selected users
  - Can also audit any other auditable event or system call
- This is in addition to audit events specified in */etc/sec/audit_events*
  - i.e., you don’t need to specify login/logout in *audit_events* in order to audit them with u_auditmask

Per-User Auditing Example

- Step 1: Edit */etc/sec/audit_events* and trim to just the items you want
  - Minimum: retain “Audited trusted events” section
- Step 2: Set up auditing (*sysman auditconfig*)
- Step 3: For each user to be audited, define per-user audited items, using *dxaccounts* or *edauth*
  - Option 3a: In *dxaccounts,* double-click username
  - Click “Security Options”
  - Select “Audit Events” from “Turn To” pull-down menu
  - Select events to audit for user and whether to audit successes, failures, or both
  - Click “OK” twice

Per-User Auditing Example (2)

- Option 3b: “edauth <username>”
- Add u_auditmask field to entry, e.g.:

```
u_auditmask=login\:1\:1,logout\:1\:1:
```

  - This audits login/logout successes and failures

- To test per-user auditing:

```
# /sbin/init.d/audit stop
# /sbin/init.d/audit start
```

- Login as the user, either sucessfully or unsuccessfully.

```
# auditd -d      (dumps audit data to the log)
# audit_tool ./auditlog.host.XXX > audit.out
```

- View the audit.out file.

Sample Output from Example

```
  :
event: login
login name: test
  :
devname: /dev/pts/7
..........
-- remote/secondary identification data --
hostname: localhost
...........
char param: Login succeeded
  :
timestamp: Mon Mar 22 09:26:09.03 2004 EST
```

Secure console mode

- Actually two modes – one software, one firmware
- Firmware: set console password and “SECURE” console variable (available on most Alphas)
  - Allows only regular boot from default device
- Software: SECURE_CONSOLE variable in */etc/rc.config*
  - YES: requires root password to enter single-user
  - NO: enters single-user mode without password
  - Not set: depends on SECURE console variable
- For more details, see *sulogin*(8) man page

HP WORLD 2004

Solutions and Technology Conference & Expo

Co-produced by:

interex

shared knowledge • shared power

encompass

AN HP USER GROUP

RECOMMENDED TRAINING VENUE FOR THE

HP Certified Professional