SCL System Administration u1

Part 3 (Update 2)

1. Securing SUSE Linux

AppArmor Facility

- Open Source Application Security System (Kernel Security module)
  - o associates a security profile with each application
  - o protects OS and Applications
  - o protects data from unauthorized users
  - o protects OS from applications violating its profile
- YaST based toolset and console used to automate security policy development
- Identifies and captures application errant behavior to create a security policy based on profiles
- name/path based mandatory access control method
- Auditable application access to resources, privileges

AppArmor Facility 2

- AppArmor implements name-based mandatory access controls
- Confines each program to a set of listed files and Posix 1003.1e capabilities
- Profiles are in learning (complain) mode or enforcement (confined) mode
- Creating a profile for an complex App is iterative
- See examples in: <https://en.opensuse.org/SDB:AppArmor_geeks>

SELinux Facility u1

- All system calls denied by default, unless specifically enabled
- All objects [files, ports, processes, users, programs, directories] have security label with User, Role and type part in the context.
- Rules show which source context has access to which target context
- Security policy customized to your system: labels for all files, services and users. (very Labor intensive)
- 3 Components:
  o Security framework in the Kernel
  o SELinux libraries and binaries
  o SELinux policy

SELinux Facility 2 u1

- The Policy: (Use ls -Z to show a directory's security context of its files; also netstat --Zutlpen, ps -Zaux)
- sample rule: **allow usr_t bin_t filename {read execute getattr}**
- 3 Modes of operation: Enforcing, Permissive, Disabled
  Set in GRUB when booting.
- Support Status:
  o Full binaries and Kernel
  o No SLES policy yet; expected in 12 SP1
  o OpenSUSE 13.1 policy works well
- Reference: Sander van Vugt Presentation <https://susecon2014.smarteventscloud.com/connect/sessionDetail.ww?SESSION_ID=7986>

AppArmor vs SELinux

- Both Supported. AppArmor uses names; SELinux uses inodes
- Mutually incompatible, neither on by default
- See Comparison Table: <https://suse.com/support/security/apparmor/features/selinux_comparison.html>
- SELinux = wide support, harder to configure, assumes all denied except…
- AppArmor = default profiles available, easier to create new configurations

2. su and sudo Commands

su sudo visudo /etc/sudoers

su Command

- **su (1)** meaning substitute userid is used by logged in users to run commands with the privileges of another user account.
- The password request is for the target userid, not the typist's.
- When executed, it can invoke a shell without changing the current working directory and/or the user environment.
- if option '**-**' or '**-l**' is specified, a login shell starts for the target user.
- if no userid is specified, the root user is assumed
- Running a single command: $ su -c {root only command}

sudo Command

- **sudo (1)** [substitute user do] is a program that lets users run programs (commands) with (elevated) security privileges of another user (usually root)
- Form: **sudo [-l] [other options]** {*Linux Command*}
- Users must provide their own password for authentication, not the root password.
- Once authenticated, if the configuration file (/etc/sudoers) permits the user access, the system runs the command.
- Logging in as root without its password: $ **sudo su -**
- **sudo** has a timer set to allow several commands before it asks for a password again

visudo Editor U2

- Required Special Editor for the sudo configuration file /etc/sudoers
- On SUSE, the target user (root) password is always requested.
- Only root is resident in the sudoers file
- In sudoers, look for uncommented lines like:

```
Defaults   env_reset
Defaults   always_set_home
Defaults   secure_path=”/usr/sbin:/usr/bin:/sbin:/bin”
Defaults   targetpw
ALL        ALL=(ALL) ALL
root       ALL=(ALL) ALL
```

- The root line is an instance of
  WHO FROM_WHERE=(AS_WHOM) WHICH_COMMANDS
- Because of **Defaults targetpw**,
  $ sudo su - # needs an explicit root password

sudo Exercise U2

- 1. Login as root and type: # **visudo**
  2. Comment lines: Default targetpw
     ALL ALL=(ALL) ALL
  3. Add Command alias:
     Cmnd_Alias NETWORK = /usr/sbin/wicked, /sbin/ip
  4. To let group 'users' members run wicked and ip, Insert line:
  %users ALL = NETWORK
  5. Open a shell as a user. Type **sudo /sbin/ip addr show**
  6. open **visudo** and include these lines:
     Cmnd_Alias NSHELLS=/bin/sh, /bin/bash
     Cmnd_Alias NSU==/bin/su
     %wheel ALL=ALL, !NSHELLS, !NSU
  7. Create user **jane**: **# useradd -m -d /home/jane jane;**
     # usermod -A wheel jane #SLES 11.3 only
     # usermod -aG wheel jane #SLES 12.x only
  8 . Login as **jane** and try running: **sudo -i**. Note it doesn't give you a root shell. but access to **visudo** is still permitted...

3. SUSE Linux Audit Framework

SUSE Linux Audit Framework u1

- Allows setting up the system for logging detailed messages using the **auditd** daemon
- Comprehensively logs and tracks access to files, directories, and resources of your system
- Traces system calls
- Monitors system for application misbehavior or code malfunctions.
- Creates a sophisticated set of rules including file watches and system call auditing
- Insures any violation of your security policies is noted and properly addressed.

SLES 11.x Linux Audit Framework (LAF) (1)

- **Set up Procedure**
  1. Stop the default audit daemon with the **rcauditd stop** command.
  2. Adjust the system configuration for audit and enable audit (for next boot)
  auditctl -e {1|0} # 1=enable, 0=disable, current session only
  For all future sessions, in /etc/sysconfig/auditd, set variable AUDITD_DISABLE_CONTEXTS = yes|no

SLES 11.x Linux Audit Framework (LAF) (2) U1

- **Set up Procedure (continued)**

  3. Configure the audit daemon. (**/etc/audit/auditd.conf**). Most defaults are OK. (e.g. change **num_logs=7**, to have two for the weekend also). See **# man 5 auditd.conf**)

  4. Determine which system components to audit and set up audit rules. 3 types: • Basic audit system parameters • File, directory watches • System call audits. Audit rules file in **/etc/audit/audit.rules** :
  (use exact file names better than directories, no wildcards)

  -D {discard old rules} , **-b 8192** {busy system}, **-f 1** {print failure mesg}

  -e 1 { 0 | 1 | 2 means 0=disable, 1=enable, 2=enable + lock down configuration }

  -w /var/log/audit/ -k LOG_audit # key tag for log file
  -w /etc/audit/audit.rules **-p rwxa** # watch this file
  -w /etc/audit/audit.rules **-p rwxa** # watch this file
  -a entry, always **-S umask** # umask system call rule

SLES 11.x Linux Audit Framework (LAF) (3) U1

- **Set up Procedure (Continued):**
- **5.** Optionally configure plug-in applications you intend to use with the audit dispatcher.
- **6.** Start the audit daemon after you have completed the configuration of the audit system using the **rcauditd start** command.
- **7.** Determine which reports to run and configure these reports with **aureport**. Options: **-i** (base 10 numbers and units) { **--summary**, [**-f**]**--success**, [**-f**]**--failed**, **-l, -p, -f, -u, -ts, -te** }

SLES 11.x Linux Audit Framework (LAF) (4) U1

- **Set up Procedure (Concluded):**
- 8. Analyze the audit logs and reports using **ausearch**. options:
  -a audit event id **-ul** login-id **-k** key **-m** mesg type
  -f filename **-p** process id, {**-ts, -te**} **-i**
- **9.** (Optional) Analyze individual system calls with **autrace**.
- See Linux Audit Quick Start <https://www.suse.com/documentation/sles11/singlehtml/audit_quickstart.html>

PAM (Pluggable Authentication Module) 1

- PAM provides authentication between the user and the application.
- Ingredients:
  - **PAM modules** (shared libraries per authentication mechanism)
  - **module stack**
  - **PAM aware service needing authentication** (e.g. login, su)
  - **Module arguments** for auth. flavor
  - **a way to evaluate each result of a single PAM module execution** (e.g. no influence, proceed or terminate immediately)

PAM (Pluggable Authentication Module) 2

- [Auth. related Programs] --use--> [**libpam.so, libpam_misc.so**] --point to **config files** in --> [**/etc/pam.d/***] -- include **plugins** in --> [**/lib64/security/***]
- PAM configuration files show 4 phase authentication process:
  auth: (initialize authentication)
  account: (check account settings)
  password: (check password settings)
  session: (constraints after successful authentication)

PAM (Pluggable Authentication Module) 2

- A PAM file can be called in different ways:
  required: conditions of this PAM library file must be met, else continue but access will be denied
  requisite: conditions of this PAM library file must be met, else stop immediately
  sufficient: conditions of this PAM library file may not be met, but if they are, any additional files do not have to be processed, else they do. {e.g. login via network, else login locally}
  optional: This is used, for example, to display a warning message regarding only authorized logins on this server.
  include: links to another PAM configuration file.
- See <https://www.suse.com/documentation/sles11/singlehtml/book_security/book_security.html#cha.pam>

PAM Exercises Part 1

- 1. Open a root shell and type:
  # **ldd $(which su)** # what libraries are used by su?
  (verify libpam.so and libpam_misc.so)
  2. Type: # **cat /etc/pam.d/su** # show su's PAM config file
  3. This config file output has common files: common-auth
  4. # **cat /etc/pam.d/common-auth**
  5. Use # **ls /lib64/security** # shows all the PAM modules
  6. Type: # **less /usr/share/doc/packages/pam/Linux-PAM_SAG.txt** # accesses PAM administrator guide

PAM Exercises Part 2 U2

- 1. type the key sequence: **<Ctrl-Alt-F4>**
  on the login prompt, login in as an ordinary user (e.g. sharon)
  2. type $ **su -** # to become root
  3. Once successfully logged in, type exit twice to log back out of both sessions.
  4. Open a root shell and type:
  # **vim /etc/pam.d/su**
  5. After the first line and before the line with **common-auth** add the line: **auth required pam_securetty.so**
  6. Open the file **/etc/securetty** in vim editor and remove the line containing tty4, if it is there.
  7. Repeat steps 1 and 2. It should not work this time.

4. Monitoring Log Files

/var/log syslog messages

SUSE System Log Protocol

- On SLES 11.3, **syslog-ng** runs (to be replaced by **rsyslog** and **journald** (has syslog functionality) in SLES 12.x.
- It is a highly portable open source protocol replacing and extending **syslogd**
- It extends timestamps to millisecond granularity and timezone information, being able to track the path of a given message, uses reliable TCP, encrypted log storage.
- **syslogd** and **syslog-ng** can coexist if installed at the same time. Then **syslogd** would handle local messages and **syslog-ng** would handle everything else.
- Communication would be via a named pipe between the two daemons.

syslog-ng

- Configuration file: **/etc/syslog-ng/syslog-ng.conf**
- This file can filter the different subsystems to different file logs
- If a single log file is desired, put this at the top of the **syslog-ng.conf** file prior to the filter definitions:
  destination catchall { (file(/var/log/catchall); };
  log { source(syslog); destination(catchall); };
- PID of running daemon stored in **/var/run/syslog-ng.pid**
- **See: <softpanorama.org/Logs/Syslog_ng/index.shtml#Recommended%20Papers>**

Message Facility.Priority

- **Facility Areas: * [**all except **mark], auth [Security, authorization** related **], authpriv [Sensitive, private authorizations], cron [crond], daemon [**system daemons**], ftp [ftpd], kern [kernel], local[0-7] [** 8 local message flavors **], lpr [** print spooler**], mail [(send)mail** related **], mark [** time stamps at regular intervals**], syslog [daemon], user [** user processes **]**

- **Message Severity Levels: emerg** [panic situation], **alert** [urgent], **crit** [Critical], **err** [other error], **warning** [warning], **notice** [unusual event], **info** [informational], **debug** [for troubleshooting]
- Putting it together: **Facility.Level. Facility1,Facility2.Level, Facility1.Level1;Facility2.Level2, *.Level, *.Level;badfacility.none**

/var/log directory

- Viewing log files:
  $ **lastlog** # aulastlog similar
  $ **pam_lastlog**
  $ **evince** # Gnome document viewer
  $ **gnome-system-log**
- Manually append messages to syslog-ng
  $ **logger** # also **vlogger**
- **logger** Options: **-i** [log the PID of the logger process each line]
  -s [also send message to STDERR]
  -f file [issue message to full path, pre-existing file]
  -p facility.priority [default is user.notice]
  -t tag [insert tag every line]
  message [if missing and no **-f**, then use STDIN]

logrotate Log Files

- Rotate the log file when file size > n
- Continue writing messages to new log file after rotating the old one (via copytruncate)
- Rotation rate (yearly, monthly, weekly, daily)
- Specify compression option for rotated files (via compress [gzip], compresscmd /bin/bzip2 and compressext .bz2)
- Rotate old log files with embedded date (via dateext)
- Execute custom shell scripts immediately after log rotation (via **postrotate scriptname**)
- Remove older rotated log files (via maxage <digit> {days})
- If missing log file, don't return an error (via missingok)

logrotate Log Files 2

- Files used:
- **/usr/bin/logrotate (1)**
- **/etc/cron.daily/logrotate (8)**
- **/var/lib/logrotate.status [default]**
- If log config files are edited (/etc/logrotate.conf), restart the **syslog-ng** daemon
- For post-OS Installed packages, their logrotate files go in **/etc/logrotate.d**

journald (SLES 12.x)

- Along with systemd-journald, this double daemon is a system service that collects and stores logging data.
- It receives: • Kernel log messages (kmsg),
  • Simple system log messages (rsyslog),
  • Structured system log mesgs via native Journal API
  • System Services STDOUT, STDERR
  • audit records, via auditd
  • collects metadata for each log message (See **systemd-journal-fields (7)**)
- Stores data in /run/log/journal [reboot empties it]. Better to store in /var/log/journal/
- See **journald.conf (5)** for configuration information located in /etc/systemd/journald.conf

Schedule recurring jobs with cron

- **cron** enables automated system maintenance (e.g. log rotation)
- Each user has available a **crontab** file owned by them in /var/spool/cron/tabs/ (and /etc/crontab for root)
- It is edited via **crontab**, **NOT vim.**
  $ crontab -l # list contents
  $ crontab -e # edit contents
  $ crontab -r # remove your crontab file
- Format of file:
  min [0-59] **hr** [0-23] **DoM** [1-31] **Mth** [1-12] **DoW** [0-7] {0 = 7 = Sunday} **full path command**
  15 09 14,28 * * /home/katz/myscript

crontab Exercise

- 1. login as normal user (sharon)
  2. Type **crontab -e**
  3. Type this *tab* separated line:
     */5 * * * * logger hello,.
  4. Save and quit (**:wq**)
  5. Wait 5 minutes. Then type:
     sudo tail -f /var/log/messages
  6. When you see your message, type
     <Ctrl-C> to exit
  7. run **crontab -r** # to delete crontab

5. Performance Monitoring & Optimization

top Performance Monitoring

- **top** (Table of Processes) gives an ordered list of running processes by user-specified criteria and updates in in real time. [**?** or **h** = help] [**q** or **<Ctrl-C>** = quit]
- It shows cpu usage (default), memory, processing power
- It shows which users and processes are consuming the most system resources at any tie.
- Load average numbers = sum of waiting processes+now executing processes
- Interactive commands: **k** = kill PID, **r** = renice PID, **d | s** = change delay time interval (seconds)
- **T**=show Load Ave/Uptime; **m**=show memory/Swap use; t=show Task/Cpu states, 1=show Single/Separate CPU states

vmstat Performance Monitoring

- **vmstat (8)** reports virtual memory statistics: about processes, memory, paging, block IO, traps and cpu activity.
- The first report gives averages since the last reboot. Others give information based on a sampling period of length “delay time”.
- Reports help identify system bottlenecks. **vmstat** itself is not part of the statistical output.
- Files used: /proc/meminfo, /proc/stat /proc/*/stat
- $ **vmstat 1 5** # delay=1 second, do 5 times, then quit
- See: <https://www.thomas-krenn.com/en/wiki/Linux_Performance_Measurements_using_vmstat>

vmstat Performance Monitoring 2

- Procs: **r**=# pids waiting; **b**=# pids sleep
- Memory: **swpd**=virtual memory used; **free**=amount of idle memory; **buff**=amount of buffer memory; **cache**=amount of cache memory; **inact**: amount of inactive memory; **active**: amount of active memory

vmstat Performance Monitoring 3

- Swap: **si**: swap in memory from disk;
  so=swap out memory to disk
- IO: **bi**: Blocks received from a block de vice (blocks/second);
  bo: Blocks sent to a block device
- System: **in**: No. of Interrupts/second
  cs: No. of context switches/second
- CPU: % of total CPU time. **us**=user time+nice time;
  sy: system time; **id**: idle time; **wa**: Wait time for IO;
  st: time stolen from a virtual machine

iostat and iotop Commands

- **iostat (8)** Gives information about the number of Blocks that was read and written on all Block oriented devices.
- use -x option for wider, better information including read ahead and write ahead gains
- **iotop** not initially installed. Use: # **zypper install iotop**
- **iotop** shows busiest process on top with its reads and writes. Processes in [ ] are kernel processes.

Network Performance Commands

- # **ip -s link** # check No. of packets sent and received and no errors
- # **ethtool eno1** # see settings of network board, make sure supported link speed can be achieved
- # **IPTraf-ng** # start it by typing **iptraf-ng.** Analyze network traffic from a menu

Optimizing Performance

- Before changing your system, create a baseline of how it currently performs
- **/proc/sys** has kernel parameters that can be changed while it's running
- Example: Change "**swappiness**" [0-100], default=60
  # echo "30" > /proc/sys/vm/swappiness # this time
- If desired permanently, edit the /etc/sysctl.conf file and set vm.swappiness=30
- Alternate change: **# sysctl -w vm.swappiness=30**
- If sysctl.conf file updated, activate via:
  # sysctl -p /etc/sysctl.conf; sysctl -a | grep vm.swap

Performance Test

- Create a 1 GB file via:
  # dd if=/dev/zero of=/root/1GBfile bs=1M count=1024
- Measure the time it takes to copy it:
  # time cp /1Gfile /tmp
- Run it again 10 seconds later:
  # time cp /1Gfile /tmp
- Why is it slower?
  # free -m # shows ~2GB cache to slow down cp

Performance Tuning

- **CPU Tuning**: (Single or Multiple CPUs) Symmetric Multiprocessing (SMP) Kernel used and load balancing required. Strive for Processes started in 1 CPU remain there once swapped back in to maintain cache. See **taskset {hex CPU: 0x[0248] }**

- **Memory Tuning**: Page size (usually 4K, unless huge files, can increase size up to 64K in powers of 2), Read and Write Cache sizes. overcommit-memory {0,1,2} overcommit-ratio {1-100}
- **Interprocess Communication**: shared memory especially for databases. See **ipcs -m**

Performance Tuning (2)

- **Storage Tuning**: journaling optimization, I/O buffer performance, I/O scheduler
- **Network Tuning**: Network Card, TCP/IP protocol stack, Application stack. Tune in that order to see effect.

Performance Tuning (3)

- Network Optimization strategies:
  - Have the latest network driver modules
  - Check Ethernet config. settings: frame size, MTU, speed, duplex mode
  - Insure all network communication devices use same settings
  - Use 9000 byte jumbo frames for fewer packets, reduced overhead

Using Control Groups U2

- Control Groups (**CGroups**): Requires libcgroup-tools RPM package; uses cgconfig and cgred services [put in run levels startup]
- When services running, **/cgroup** directory contains subdirectory controllers.
- Useful controllers are:
  - **blkio** limit the amount of I/O that can be handled
  - **cpu** limit CPU cycles
  - **memory** limit grantable memory to processes
- See: <https://www.suse.com/documentation/sled11/book_sle_tuning/data/sec_tuning_cgroups_usage.html>

Memory, Storage, Network Performance

* Reference: <cs.cornell.edu/projects/ladis2009/talks/dean-keynote-ladis2009.pdf

- Numbers Everyone Should Know*

Action	Time (1 ns = 10^{-9})
L1 cache reference	0.5
Branch mispredict	5
L2 cache reference	7
Mutex lock/unlock	25
Main memory reference	100
Compress 1K bytes with Zippy	3,000
Send 2K bytes over 1 Gbps network	20,000
Read 1 MB sequentially from memory	250,000
Round trip within same datacenter	500,000
Disk seek	10,000,000
Read 1 MB sequentially from disk	20,000,000
Send packet CA-> Netherlands->CA	150,000,000 or 0.15 sec

6. System Administration Time Management

Handling Interruptions

- Unavoidable but manageable
- Being interrupt-driven means priority is fifo (first in, first out), events manage your time
- Acknowledge interrupter and either:
  - **Delegate** (to a different specialist)
  - **Record it** (if not urgent, write request down accurately, for later action)
  - **Do it** (if urgent, i.e. an outage, switch to that request)
- This lets you take (back) control of your time

Recommendations U1

- 1. Measure twice, cut once: (check again before you make a change that is not reversible)
- 2. Make a backup before you change a file
- 3. If all else fails, read the man(ual) page
- 4. When debugging a script or procedure, change one thing at a time.
- 5. Always test your work
- 6. You're not done until your customer tests it, too.
- 7. The strangest problems arise from misconfigured DNS
- 8. The most popular command problems are due to permission incompatibility

Good Routines

- Always keep your organizer/phone with you (as well as charging hw)
- Meet with your boss regularly
- If you have to ask, the answer is yes
- During outages, communicate to Management
- Use automatic checks while doing certain tasks
  [i.e. audible ping:
  $ **ping -s IPaddress | tr : <Ctrl-G> <Ctrl-G>**
- Always back up a file before you edit it.
- Make routines for:
  • maintenance tasks • things you forget often • Developing new skills • Keeping up-to-date

Calendars

- Record everything; use to guide your day
- Use for:
  - Appointments and Meetings
  - Milestones, important dates
  - Future to-do items
- Call if late or need to cancel
- Schedule private time

Stressors

- Overloading (prioritize workload, get more sleep, explain to someone/ something else, take vacations)
- Handling Conflicting Directions (have your boss(es) prioritize your tasks, reduce bosses to (1)

Email Management

- To keep your inbox clear or to a minimum
- Filter (Use procmail)
- Delete unread mail
- Read and ... Delete
  File
  Reply then delete
  Delegate, forward, delete
- Do now then Delete
- Pick a too old date, name folder too-old.date, move old mail into that folder
- If not accessed in a year, burn folder on CD, delete folder

Automation

- Things done only once (HW/SW installations) - delegate by outsourcing
- Ongoing repeatable tasks (document with detailed procedural diary)
- Scale up: Prefer Automated OS Installations to individual ones
- Write collaborative documentation (twiki.org) which is a dated and a living document
- Is your process automated enough? Yes, if you can delegate it and it gets done successfully

Automation Examples

- Keep a Linux SA diary:
  $ echo alias diary='cd ~/documents/SAdiary/ && date >> d.log' >> ~/.bashrc
- Connect to frequently accessed remote server
  $ echo alias RK='ssh u68732801@arkaye.com' >> ~/.bashrc

$ echo alias RKftp='sftp u68732801@arkaye.com' >> ~/.bashrc

- Update bash with new contents of .bashrc
  $ ./.bashrc # or source ./.bashrc

Automation Examples 2

- Keep ssh host names in **~/.ssh/config**

```
Host dev
    HostName dev.example.com
    Port 22000
    User fooey
Host github.com
    User git
    IdentityFile ~/.ssh/github.key
Host tunnel
    HostName database.example.com
    IdentityFile ~/.ssh/katz.example.key
    LocalForward 9906 127.0.0.1:3306
    User katz
```

- Use: $ **ssh dev**
  $ **ssh -f -N tunnel**

Use Makefiles

- Keep track of Application Config files

sendmail	**/etc/aliases**	**run newaliases after changes**
postfix	**/etc/transports**	**run postmap transports " "**
m4	***.m4 files**	**run m4 after changes**

- Sample makefile (tab delimited):

```
all:    aliases.db  access.db
aliases.db:      aliases
    newaliases
    @echo Done updating /etc/aliases
```

- Run: $ **make aliases.db** # to run that section of the makefile

Automating Big Commands

- Command to detect whether excessive ARP packet requests are occurring for your machine's Ethernet (MAC) address:
- $ **sudo tcpdump –l –n arp | grep 'arp who-has' | **
  head -100 | awk '{ printf $NF}' | sort | uniq –c | \ sort -rn
- **tcpdump** listens to the local Ethernet (**-l** = enable pipelining output), (**-n** suppress DNS address lookups for IP addresses output), (**arp** means only display arp packets) # insure you are permitted to look at arp packets
- The **grep** command is extracting those lines that display:
  arp who has "IP address1" tell "host IP address2"
- Show 1st hundred lines of this. Further filter to extract:
  host IP address2 and alphabetic sort, count duplicates and sort the counted numbers from highest to lowest.
- On a non-infected Server, it may take days output 100 output lines. On Worm/virus infected server it could take a minute.

The Last Slide

- Thanks for your attendance and attention!