

# LPIC-1 102-500 – Lesson 17

## 110.1 Perform security administration tasks



# The SUID/SGID flags

- The **SUID** and **SGID** flags are used to give the right to normal users to execute a command with the rights of another user or group.
- `-rwsr-xr-x 1 root root 90640 2011-08-09 19:16 /bin/mount` # the `/bin/mount` binary can be executed with **root** rights by any user in the system.
- `-rwxr-sr-x 1 root shadow 50760 2011-06-24 12:28 /usr/bin/chage` # the `/usr/bin/chage` binary can be executed with **shadow** group rights by any user in the system.
- `-rwsr-sr-x 1 daemon daemon 47848 2011-05-16 13:32 /usr/bin/at` # the `/usr/bin/at` binary can be executed with **daemon** user and group rights by all users.



# Security concerns of SUID/SGID

- The presence of the **SUID/SGID** flags on binaries may be convenient but imposed very serious security risks if the command has some security vulnerability (e.g. buffer overflow).
- A vulnerable command can give the opportunity to an attacker to use it in a way it was not purposed. Imagine for example if the **mount** command could call the **bash** shell! This would mean that **bash** would have the same rights of the SUID user of **mount** which is **root**!
- For this reason we need to check our system for SUID/SGID flags and avoid setting it to commands that can modify files or call the shell, like **vi** or **emacs**.

# Find and remove SUID and SGID

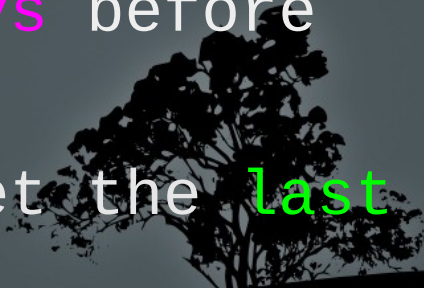
- `# find / -perm -4000 -type f -ls # check all normal files for the SUID flag, everywhere in the system.`
- `# find / -perm -2000 -type f -ls # check all normal files for the SUID flag, everywhere in the system.`
- `# chmod u-s /bin/ping # remove SUID from /bin/ping`
- `# chmod g-s /usr/bin/crontab # remove SGID from /usr/bin/crontab`
- `# chmod -s /usr/bin/at # remove SUID and SGID from /usr/bin/at`



# The */etc/shadow* file

- The **/etc/shadow** file contains the hashed passwords of the users but also useful information about the expiry of passwords. The **/etc/shadow** fields are assigned the following roles:
  - **user:\$6\$UwkipSFw\$Jp3JxkKjZJ48zdM:15428:5:20:7:15:15695:**
    1. **Username**
    2. **Hashed password (! or nothing: no password, \*: disabled account, ! <hash>: locked account. !!: password not set)**
    3. **Last change date**
    4. **Minimum number of days where the user can change the password (0 means the passwords can be changed any time).**
    5. **Maximum number of days where the user can keep the same password (99999 means no change required)**
    6. **Number of days before the expirations where a warnings will be issued.**
    7. **Number of days (inactivity days) after the expiration where the account will be disabled.**
    8. **Expiration Date (number of days after 01/01/1970)**

# Manage password expiry information with `chage`

- `# chage -l user1 # show expiry information for user1.`
  - `# chage -E 2012-12-21 user1 # set expiration date.`
  - `# chage -I 15 user1 # set inactivity days.`
  - `# chage -m 5 user1 # minimum days between password changes.`
  - `# chage -M 20 user1 # maximum days during which a password is valid.`
  - `# chage -W 6 user1 # warning days before expiration.`
  - `# chage -d 2012-03-25 user1 # set the last password change date.`
- 

# Using `date` for showing change/expiry dates

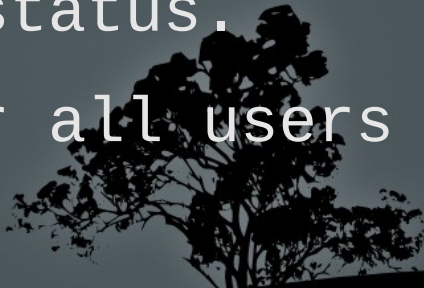
- `user:$6$UwkipSFw$Jp3JxkKjZJ48zdM:15428:5:20:7:15:15695:`
- `# date -d "1970/01/01 +15428 days"`  
Thu Mar 29 00:00:00 EEST 2012 # last pass change date
- `# date -d "1970/01/01 +15695 days"`  
Fri Dec 21 00:00:00 EET 2012 # password expiry date

These values can be used as parameters in the `chage` command to set days after 01/01/1970 (unix epoch)

- `# chage -d 15428 user # = chage -d 2012-03-29 user`
- `# chage -E 15695 user # = chage -E 2012-12-21 user`



# Using `passwd` to manage expiry information

- # `passwd -i 15 user1` # set **inactivity days**.
  - # `passwd -n 5 user1` # **minimum days** between password changes.
  - # `passwd -x 20 user1` # **maximum days** during which a password is valid.
  - # `passwd -w 6 user1` # **warning days** before expiration.
  - # `passwd -e user1` # force password expiration and prompt for password change.
  - # `passwd -S user1` # show **user1** status.
  - # `passwd -Sa` # show statuses for all users
- 




# Using `usermod` to manage expiry information

- `# usermod -e 2012-12-21 user # set expiration date.`
- `# usermod -f 15 user # set inactivity days.`
- `# usermod -L user # lock account.`
- `# usermod -U user # unlock account.`



# Detect open ports on the system

- Ports on a system are used to provide access to applications “listening” to them.
  - Sometimes systems have pre-installed services that may not be needed.
  - It is a good practice to disable unused services to save resources but most importantly to minimize the attack surface on a system. Attackers may use existing vulnerabilities in these services to penetrate the system.
  - To check for open ports we can use the **ss**, **netstat**, **lsof** and **nmap** tools.
- 

# Check for open ports with `ss` and `netstat`

- The `ss` and `netstat` commands can be used to show the open ports on a system.
- `# ss -lnptu` # show all listening tcp and udp ports in numeric format and the programs that use them.
- `# netstat -lnptu` # show all listening tcp and udp ports in numeric format and the programs that use them.



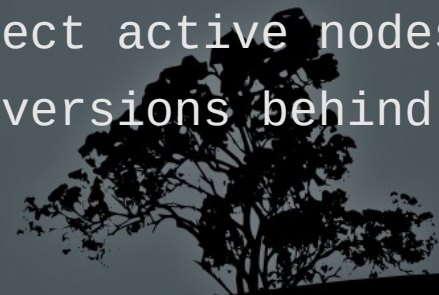
# Check for open ports with `lsof`

- The `lsof` command is used to display open files in the system. Sockets and ports are also considered files in a Linux system.
- `# lsof #` show all open files in a system.
- `# lsof -i #` show all TCP/IP connections and ports.
- `# lsof -iTCP -s:LISTEN -P #` show all TCP listening ports in numeric form (-P).
- `# lsof -iUDP | grep -v "\->" #` show open UDP ports
- `# lsof -p 6543 #` show all open files of the 6543 process.
- `# lsof -c apache2 #` show open files of all apache2 processes.
- `# lsof -u user1 #` show all open files by `user1`.
- `# lsof /mnt #` show processes using the `/mnt` directory.



# Detect open ports with `nmap`

- Unlike the `ss`, `netstat` and `lsof` commands, `nmap` can detect open ports on other computers. In some countries its use is forbidden. It is a good practice to use it only on computers you own.
- `$ nmap sT www.network.dom # (TCP Connect scan) default type of scan for non-privileged users.`
- `# nmap -sS www.network.dom # (TCP Syn scan) default type of scan for non-privileged users (faster).`
- `# nmap -p 65-87,100 www.network.dom # check ports 65 to 87 and 100 (TCP).`
- `# nmap -p 1-65535 -O www.network.dom # check all ports and detect operating system.`
- `# nmap -sU -n www.network.dom # UDP scan with numeric presentation.`
- `# nmap -sP 10.0.0.0/24 # ping sweep to detect active nodes.`
- `# nmap -sV 10.0.0.3 # detect services and versions behind open ports.`



# Switch users with `su`

- The **su** command is used to login into the system as another user. You have to use **the other user's password to login**. If no user is defined **root** is implied.
- `$ su #` login as **root** inheriting the environment of the original user.
- `$ su - #` login to the system as **root**. The environment will be the same as if we login directly as root (switch to the home directory, run **.bash\_profile** or **.profile** etc)
- `$ su user1 #` switch to **user1**
- `$ su - user1 #` switch to **user1** in an environment same as **login**.
- `# su - user #` the root user can assume the role of any other system user without using a password!
- `$ su -c "find /etc" #` run the **find** command with **root** privileges.

# Run commands as another user with `sudo`

- The **sudo** command is used to execute a command as another user but **using your own password**. For this to happen the user that needs **sudo** command execution rights must be declared in the `/etc/sudoers` or belongs to a group that is declared in said file.
- `$ sudo systemctl restart ssh # run command as root.`
- `$ sudo -u user1 mail # run command as user1.`
- `$ sudo -i # run a bash shell as root.`
- `$ sudo -b updatedb # run a background command as root.`

# Configuring sudo with */etc/sudoers*

- In the */etc/sudoers* file we declare all users or groups that have the right to use **sudo**. It's a read-only file so it is not recommended to be edited by any other text editor besides **visudo**.
- `# visudo # open /etc/sudoers for editing`
- `user1 ALL=(ALL) ALL # give the right to user1 to run on any system, as any user χρήστης, any command.`
- `user1 mypc = (operator) /usr/bin/mount, /bin/kill, /usr/bin/lprm # give the right to user1 to run on mypc, as the operator user, the commands mount, kill and lprm.`





# Configuring sudo with */etc/sudoers*

- `user1 server = (operator : operator) /usr/bin/mount, /bin/kill # give the right to user1 to execute on server, as user and group operator, the commands mount and kill.`
- `user1 hostname = (operator) /usr/bin/mount, (root) /bin/kill # give the right to user1 to run as operator the command mount and as root the command kill.`
- `user1 ALL = NOPASSWD: /bin/kill, PASSWD: /usr/bin/mount # give the right to user1 to run as root, the command kill without a password and the command mount using a password.`
- `%admin ALL=(ALL) ALL # assign all rights to the admin group.`



# Set limits with the */etc/security/limits.conf* file

- In the */etc/security/limits.conf* file we define the limits for the different resources of the system.
- Its format is:  
    <domain>      <type> <item>      <value>
- **domain:** **username** (user1), **group** (@group1) or **\*** (everybody)
- **type:** **soft** (soft limit), **hard** (hard limit), **-** (both). The soft limit can be exceeded by the users using the **ulimit** command while the hard limit can not.
- **item:** set the resource to limit e.g. **maxlogins**, **nproc**, **cpu**, **memlock**, etc.
- **value:** the limit value. It can be in kB for data resources, or minutes for time resources or even just a number of files, resources etc.



# Set limits with the */etc/security/limits.conf* file


- Example items:
  - **maxlogins**: maximum sessions number.
  - **nproc**: number of processes.
  - **stack**: stack memory size.
  - **memlock**: locked memory size.
  - **as**: memory space size.
  - **cpu**: CPU usage time.
  - **fsize**: files size.
  - **nofiles**: number of files.



# Set limits with the */etc/security/limits.conf* file

- Example of limits in */etc/security/limit.conf*:

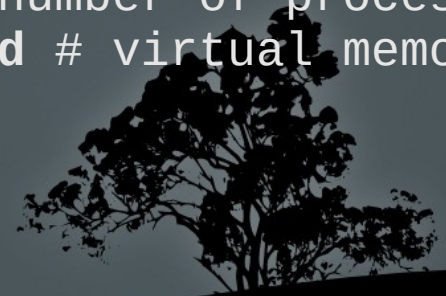
<domain>	<type>	<item>	<value>
*	hard	memlocks	10000
@student	hard	nproc	20
@faculty	soft	nproc	20
@faculty	hard	nproc	50
ftp	hard	nproc	0
@student	-	maxlogins	4

- The hard limit for **memlocks**, for all users is **10000 kB**.
  - The hard limit for **nproc** for members of **student** is **20** processes.
  - The soft and hard limit for **nproc**, for the members of the **faculty** group is 20 and 50 processes respectively.
  - The **ftp** user has no right to execute processes.
  - The members of the **student** group are allowed 4 sessions each.
- 

# Set user limits with `ulimit`

- The **ulimit** command is used to temporarily change the resources of the shell we are currently working with and all its child processes.
- Only the **root** user can define limits and only the root can change its hard limit.
- Normal users can only redefine their own soft limit and it should not exceed the hard limit.

```
$ uname -a # = uname -Sa. Show soft limits
max locked memory      (kbytes, -l) 64 # locked memory
max memory size        (kbytes, -m) unlimited # resident memory
stack size             (kbytes, -s) 8192 # stack size
cpu time               (seconds, -t) unlimited # CPU time
max user processes     (-u) 30966 # number of processes
virtual memory         (kbytes, -v) unlimited # virtual memory
```



# Set user limits with `ulimit`

- `$ ulimit -Ha # show the user's hard limits.`
- `$ ulimit -u 45000 # increase the number of processes limit to 5000.`
- `# ulimit -Hs 16384 # set the hard stack limit to 16MB.`
- `$ ulimit -St 2 # set the soft cpu time limit to 2 minutes.`
- `# ulimit -v 2048000000 # increase virtual memory limits (soft and hard) to 2GB.`
- `$ ulimit -l 128 # increase the soft locked memory limit to 128kB.`

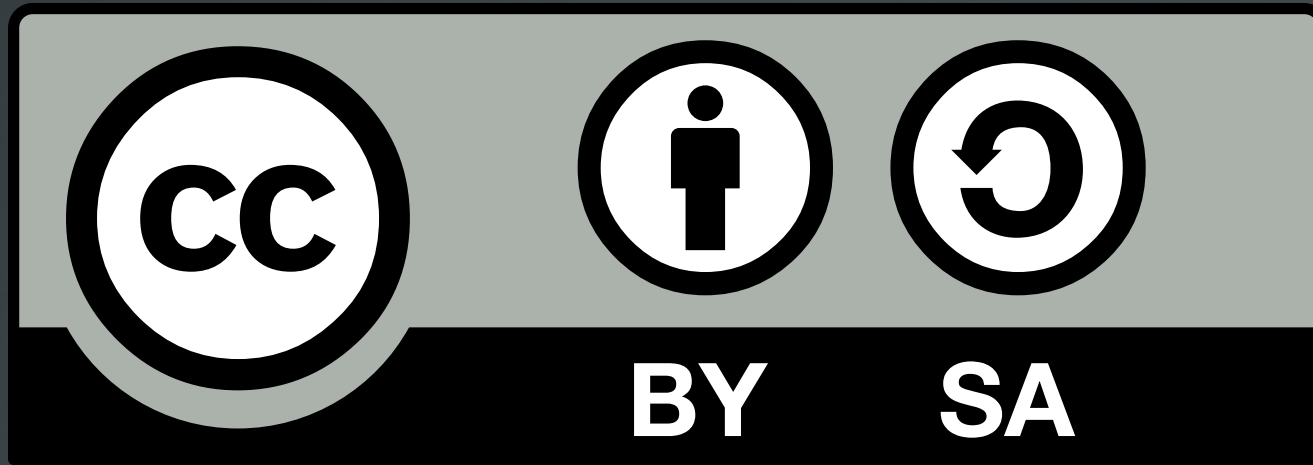


# The ``who``, ``w`` and ``last`` commands

- `$ who #` show the logged in users and sessions.
- `$ w #` a better alternative to the `who` command, showing user and sessions with more details.
- `$ last #` show the more recent logins, shutdowns and reboots.
- `# lastb #` show the more recent failed logins.



# License



The work titled "LPIC-1 102-500 – Lesson 17" by Theodotos Andreou is distributed with the Creative Commons Attribution ShareAlike 4.0 International License.

