DAT151
Database and Unix System Management

Spring 2024

Assignment 3 - Unix

Obligatory assignment. Deadline: Sunday 03.03.2024.

The report should include all necessary commands to complete the tasks, printout from the system, explanation of what is done, the result and explanation of the result.

Remember to include the names of the group members on the front page of the report. The report can be in English or Norwegian. The report should be handed in via Canvas.

The assignment should be accomplished in groups of two to three students. Signing up for a group will be closed on Thursday 29.02.

You must select a group "Lab3 N" when delivering to see all the comments that you will get on this assignment.

***Observe**: Task four will require a minimum of two lab computers, and a minimum group size of two is necessary.*

***Observe**: Before you modify an existing configuration file, save a copy of the original somewhere on the computer, e.g. below /root/origs.*

Lecturers:
Bjarte Kileng <Bjarte.Kileng@hvl.no>, room E418
Haakon André Reme-Ness <Haakon.Andre.Reme-Ness@hvl.no>, Fabrikkgaten 5, R235-02

Last changed, 21.03.2024 (BK)

Task 1: SELinux

SELinux (*Security-Enhanced Linux*) , introduced in chapter three of the Unix book, is a Linux kernel security module that provides access control security policies.

The SELinux context for a file is given as *selinux_user:selinux_role:selinux_type:mcs*. A web file can e.g. have a SELinux context of *unconfined_u:object_r:httpd_sys_content_t:s0*.

The SELinux type part (e.g. *httpd_sys_content_t*) is used by the SELinux policy rules to grant access to objects and files. The SELinux role part (e.g. *object_r*) determines the allowed SELinux types of the object, and the SELinux user part (e.g. *unconfined_u*) determines the allowed SELinux roles. For the MCS part (e.g. *s0*), see e.g. Using Multi-Category Security (MCS) for data confidentiality.

a. Check whether SELinux is currently enabled. Make sure SELinux is enabled and in targeted and enforcing mode.

b. Working with SELinux users.

   1. Check the man page of **semanage** and list the mapping between Linux users and SELinux confined users on your computer.

   2. Check the man page of **seinfo** to list all available SELinux users on your computer.

   3. Create a new Linux user, and use SELinux to prevent this user from using the **su** and **sudo** tools.

      - Hint, see e.g. SELinux user capabilities.

c. The SELinux context of a process is often named a *domain*. SELinux domains and contexts have the same structure, i.e. *selinux_user:selinux_role:selinux_type:mcs*.

   Install the Apace web server on your computer (package *httpd*).

   When Apache is started with the **systemctl** command, the web server will be started by the *systemd* process that has a SELinux type of *init_t*. Apache will run with the SELinux type of *httpd_t* and will read web content below the directory */var/www/html*.

   Since the SELinux types of *systemd* and Apache is different, the web server process has to transition from *init_t* to *httpd_t*. The actual domain transitions are triggered by the processes themselves, but SELinux policies must allow the transitions.

   Verify that an Apache web server that is started by *systemd* has read access to the content of */var/www/html*. You have to demonstrate how SELinux allows the access:

   1. Is *systemd* allowed to start the Apache web server?
      - Determine the SELinux type of *systemd*.
      - Determine the SELinux type of the Apache executable file.
      - Determine if *systemd* is allowed to run the Apache executable.
   2. Can the Apache web server run in domain *httpd_t*?
   3. Is *systemd* allowed a transition to *httpd_t*?
   4. Has domain *httpd_t* access to open and read files in directory */var/www/html*?

   For each of the steps above, you must use the command *sesearch* and document the policy rules that apply.

   When checking for access rights, the answer often replaces the SELinux type with an attribute value. The access rule then applies to all types that has that attribute. E.g. to check if type *initrc_t* has attribute *initrc_domain*, use:

```
seinfo -tinit_t -x | grep initrc_domain
```

For more information, see e.g. SELinux/Tutorials/How does a process get into a certain context.

d. Use a SELinux boolean to allow Apache to read web content in a *public_html* directory in the home directory of users.
   - Hint, install the package *selinux-policy-doc* and check the man page of *httpd_selinux(8)*.

e. Create a directory */www*, and configure SELinux to allow Apache to read web content in this directory.
   - Hint, see e.g. Configuring SELinux for applications and services with non-standard configurations and SELinux/Labels – Gentoo Wiki.

More documentation can be found here:

- Using SELinux (RHEL 9)
- SELinux User's and Administrator's Guide (RHEL 7) (Still valid, and more extensive)

Task 2: Printing

Install CUPS and all necessary dependencies, start the cups service, and then set it to run on start up. Then add the HVL printing system.

1. Send a print job from *LibreOffice*.

2. Send a print job from the command line using the lpr command to the HVL printer system.

Hint, the HVL web page for printing can help you to install the printer.

The script provided by the web page configures the printer to ask for credentials for each print job, but the **lpr** command does not support this. When logged in through a desktop environment, e.g. Gnome, the desktop environment can ask for the credentials, and **lpr** should work.

The credentials can be saved with the printer configuration. Then print jobs should then not ask for credentials, and the **lpr** command will work also in a non-graphical environment, e.g. if logged in through SSH. Be careful though not to leak the credential information, e.g. try not to write the authorization details on the command line.

Task 3: Open ports and processes

Find all ports open for **tcp** and **udp** connections on your computer, and list the processes and services that use these ports.

Task 4: 2FA

You will need a minimum of two lab computers for this task!

In this task you will set up 2FA for SSH login on your computer, using SSH key and time based one time passwords.

Observe that the file */etc/ssh/sshd_config.d/50-redhat.conf* has a configuration that collides with the settings that you will need to apply. You must therefore also modify that file.

Set up 2FA on all of the lab computers of the group. Verify that you can log in from one computer to the other using SSH, and that 2FA is used for authentication.

Observe that the **google-authenticator** secret must be stored below "*~/.ssh/*" to get the correct SELinux context. If you create the secret elsewhere, you must move the file to ""*~/.ssh/*" and then correct the SELinux context. If the context is wrong , the 2FA login will fail!

For 2FA, the SSH authentication method *keyboard-interactive* must be used, not the default *PasswordAuthentication*. On EL9 both are used with PAM, and both usually authenticates by asking for the user password. The difference is that *keyboard-interactive* allows for multiple challenge responses and the prompts are set by the server. The default *PasswordAuthentication* asks for a single password. No prompt is sent by the server, i.e. the prompt asking for the password is decided and set by the client.

The **google-authenticator** PAM module requires the *keyboard-interactive* method with SSH because:

- The login prompt asks for a verification code, not a password - server set prompt.
- The verification code can be accompanied with a prompt asking for the user password - multiple challenge responses.

For more information see e.g. RFC 4256 (*keyboard-interactive*) and RFC 4252 (*PasswordAuthentication*). See also the book SSH: The Secure Shell *The Definitive Guide* and this discussion.

Task 5: Secure your computer

Use *Fail2Ban* to secure your computer. Also set PAM requirements for password qualities, password history and account locking if many consecutive authentication failures.

Do not modify directly the PAM files. Rather modify configuration files, and configure the PAM system using **authselect** to select a PAM profile and enable or disable features of the profile.

Task 6: SSH

Create a SSH key pair as a normal user on your own computer (not your lab computer), and ask the lecturer to give you access to the lab gateway computer *eple.hvl.no*. You must provide your student number, and the public SSH key.

Observe that there is an error in SSH on Windows and the initial hand shake does not succeed to agree on a MAC that is required by eple.hvl.no. Specifying the MAC on the ssh command line should solve this, but also this fails. A solution that do work is to add the list of MACs to the config file on the Windows client.

The command below will show, among more information also the list of MACs approved by eple:

```
nmap --script ssh2-enum-algos -sV -p 22 eple.hvl.no
```

The account on eple does not have a key for your lab computer Therefore your client computer must run the SSH agent. Most Linux distributions will automatically start the SSH agent. If using Windows, see e.g. OpenSSH : Use SSH-Agent.

Start an *OpenSSH* server on your lab computer.

1. Try to log in to your lab computer from you own computer. Explain the result and why this happen.
2. From your own computer, log into your lab computer with a jump through *eple.hvl.no*.
3. Install the *MariaDB* client tool (the *mysql* shell window) on your own computer. Set up an SSH tunnel for *MariaDB* from you own computer through *eple.hvl.no* to the *MariaDB* server on your lab computer. Then use the *MariaDB* client tool on your computer to access the *MariaDB* server on your lab computer.