Linux Security

Intro

- This session is provided by Sander van Vugt
- Participants are expected to have decent Linux knowledge / experience
- To follow along, install a virtual machines running CentOS Stream (preferred) or Ubuntu Server
- If you can't use a virtual machine, feel free to use the O'Reilly Sandbox (Rocky for Red Hat family, Linux for Ubuntu)

Poll Question 1 (single answer)

Rate your Linux knowledge

- none
- poor
- average
- RHCSA/LFCS/Linux+/LPI1 certified or equivalent knowledge
- Beyond basic certification

Poll Question 2 (single answer)

What is your main professional Linux distribution?

- Red Hat family
- Ubuntu and alike
- SUSE
- Something else (please specify in group chat)

Poll Question 3 (single answer)

Where are you from?

- Middle East
- Africa
- India
- Asia (other)
- North/Central America
- South America
- Pacific region
- Europe
- Netherlands

Agenda

- Linux Security Overview
- LUKS encrypted devices and NBDE
- AIDE
- PAM and authselect
- Auditing
- SELinux
- OpenSCAP
- Optional Topics: ACLs, permissions, Firewalling, GRUB2 passwords

1. Linux Security Overview

Linux security overview

- Permissions and capabilities
- GRUB2 passwords
- LUKS encrypted storage
- sudo
- Pluggable Authentication Modules (PAM)
- Filesystem permissions and attributes
- Firewalls
- Mandatory Access Control: SELinux and AppArmor

2. Boot and Physical Security

2.1 GRUB passwords

Why use GRUB passwords?

- When a boot password is set, it needs to be entered before booting
- A GRUB edit password makes it impossible to enter GRUB boot options while booting without entering the password
- GRUB passwords make a system more secure, but don't offer protection against booting from external media

Understanding what to protect

- The GRUB boot password is a password that is required for booting the computer
- The GRUB edit password is required for editing GRUB menu entries

GRUB edit passwords on Red Hat

- On Red Hat, use **grub2-setpassword** and enter the password twice
- This will create the /boot/grub2/user.cfg file containing the username root and the hash of the encrypted password
- The password will be required for making changes to GRUB2 boot options

GRUB boot passwords on Red Hat

- To set an edit password on Red Hat, look for the boot loader specification file in /boot/loader/entries
- In this file, add the line grub_users root
- Save and close the file and reboot. You'll be prompted for the actual root user password

GRUB Passwords on Ubuntu

- To set a boot password on Ubuntu, use **grub2-mkpasswd-pbkdf2** and enter the password you want to use
- This prints the hash of the encrypted password on screen
- Next, edit the /etc/grub.d/40_custom file and add the following:

```
set superusers="root"
password_pbkdf2 root grub2.pbkdf2.sha512.10000.0CbA36....
```

- Use **grub-mkconfig -o /boot/grub/grub.cfg** to apply the changes

2. Boot and Physical Security

2.2 Encrypted Devices

Understanding LUKS

- Linux Unified Key Setup (LUKS) is the standard way for creating encrypted devices
- It encrypts a complete device, resulting in a new device manager device
- This device manager device needs to be opened, after which a filesystem can be created on that device
- After encrypting device, it's the device manager device that should be mounted, and not the original device
- To access the encrypted device, a password must be entered manually or automatically through /etc/crypttab

Demo: Creating a LUKS Encrypted Device

- **cryptsetup luksFormat /dev/sdb3**
  - Provide a strong password
- **cryptsetup luksOpen /dev/sdb3 secret**
- **mkfs.ext4 /dev/mapper/secret**
- **mount /dev/mapper/secret ...**

EX415: NBDE

- NBDE is Network-bound Disk Encryption
- It automates entering passphrases to unlock LUKS encrypted devices
- Tang is the server component, Clevis the client component

EX415: Setting up NBDE

- on the server: **dnf install -y tang**
- **systemctl enable --now tangd.socket**
- on the client, to test accessibility: **curl -f http://tangserver/adv**
- **dnf install clevis clevis-luks clevis-dracut**
- **clevis luks bind -f -d /dev/sdb1 tang '{"url":"http://tangserver"}'**
- Create /etc/crypttab:
  - secret /dev/sdb1 none _netdev
- Mount through /etc/fstab using the _netdev option

3. Authentication Related Security

Understanding PAM

- PAM is Pluggable Authentication Modules
- It helps separate the specific authentication approach from the binary that needs it
- PAM also provides modules that may be used by different binaries
- It can be used to enhance security in different environments
- PAM can be configured through different configuration files in /etc/pam.d
- In RHEL 8 and later, administrators are expected to not edit these files directly, but select and change security profiles using the **authselect** tool.
- Use **authselect select minimal** to apply the minimal security profile
- Don't manually change PAM configuration if authselect is used!
- The exam evaluation will give no points if you're not using authselect!!

EX415: Using authselect

- **authselect current** lists the current security profile and features used
- **authselect list-features minimal** lists available features
- **authselect show minimal** shows a description of a profile
- **authselect select minimal with-pwhistory with-mkhomedir** selects a profile with specific features
- **authselect enable-feature with-faillock** enables a feature for the current profile

EX415: pam_faillock

- The pam_faillock module has replaced the pam_tally2 module for blocking access after failed login attempts
- It can be integrated directly in the PAM configuration files, it can also be configured using **authselect select minimal with-faillock --force**
  - Tip: use **authselect select [Tab][Tab]** for easy access to all options
- This commands writes to /etc/pam.d/password-auth
- Further configuration goes through /etc/security/faillock.conf
- For an overview of login related events, use the **faillock** command

EX415: pwquality

- The pwquality PAM module can be configured to manage password properties
- Use /etc/security/pwquality.conf for its configuration
- All the options that have credit in the name are confusing, set to a negative number to specify an absolute number of characters
  - **lcredit=-3** means you need at least 3 lowercase characters

3. AIDE

EX415: AIDE

- AIDE works with a database that has been generated with **aide --init** to summarize the current state of files
- To detect if files have changed, this database needs to be compared with the current state of the filesystem as reported by **aide --check**
- The aide.conf file is used to specify what exactly to look for and can be used to exclude files as well
- AIDE will find changes, it won't find what has triggered the change. In order to see that as well, and audit watch rule can be added

4. Filesystem Related Security

4.1 Basic permissions

Basic Permissions

- Basic permissions are read (4), write (2) and execute (1)
- Permissions are assigned to file owner and group owner
- Use **chown** and **chgrp** to set file owners
- Use **chmod** to set file permissions
  - **chmod +x myscript**
  - **chmod 640 myfile**

4. Filesystem Related Security

4.2 Special Permissions

Special Permissions

- SUID
  - **chmod 4770 myfile**
  - **chmod u+s myfile**
- SGID
  - **chmod 2770 mydir**
  - **chmod g+s mydir**
- Sticky bit
  - **chmod 1770 mydir**
  - **chmod +t mydir**

4. Filesystem Related Security

4.3 ACLs

Understanding ACLs

- Access Control Lists (ACLs) allow administrators to grant permissions to more than one user and/or more than one group
- ACLs are supported by all modern filesystems as a default
- Use **getfacl** to show current ACL settings and **setfacl** to manage ACLs
- ACLs may be used in different situations
  - In a shared user environment, where one user or group needs full access to files, and other users or groups need read-only access
  - In a developer environment, where a developer may require access to a server document root

Managing ACLs

- Use **getfacl** to see current ACL settings
- Use **setfacl** to manage ACLs
- A regular ACL will take care of all currently existing files
- A default ACL will take care of all new files
- Use ACLs as an infrastructural solution: they should be configured on directories before you start to work with files in these directories

Demo: Managing ACLs

- **setfacl -R -m g:account:rX /data/sales**
- **setfacl -m d:g:account:rx /data/sales**
- **setfacl -x g:account /data/sales**

4. Filesystem Related Security

4.4 Attributes

Understanding Attributes

- Posix defines a number of attributes that can be used to add security to files
- Use **chattr** to set them and **lsattr** to get an overview of currently applied attributes
- Of all attributes, the immutable (**i**) attribute is common

Demo: Using Attributes

- **touch /root/removeme.txt**
- **chattr +i /root/removeme.txt**
- **rm -f /root/removeme.txt**
- **ls -l /root/removeme.txt**
- **lsattr /root/removeme.txt**
- **chattr -i /root/removeme.txt**
- **rm -f /root/removeme.txt**

5. Network Security

5.1 Linux firewalling overview

5. Network Security

5.2 iptables

Demo: **iptables** Intro

- **iptables -P OUTPUT DROP**
- **iptables -P INPUT DROP**
- **ping google.com**
- **iptables -A OUTPUT -p icmp -j ACCEPT**
- **iptables -A OUTPUT -p tcp --dport=53 -j ACCEPT**
- **iptables -A OUTPUT -p udp --dport=53 -j ACCEPT**
- **ping google.com**
- **iptables -A INPUT -m state --state=ESTABLISHED,RELATED -j ACCEPT**
- **ping google.com**

5. Network Security

5.3 Opening ports and services with firewalld

Understanding **firewalld** Ingredients

- **Zone**: a collection of network cards that is facing a specific direction and to which rules can be assigned
- **Interfaces**: individual network cards, always assigned to zones
- **Services**: an XML-based configuration that specifies ports to be opened and modules that should be used
- **Forward ports**: used to send traffic coming in on a specific port to another port which may be on another machine
- **Masquerading**: provides Network Address Translation on a router
- **Rich rules**: extension to the **firewalld** syntax to make more complex configuration possible

Working with **firewall-cmd**

- **firewall-cmd** is the default CLI for **firewalld**
- It may appear overwhelming, but is very well structured
- The elements previously listed can be managed easily, use **firewall-cmd --help** to get syntax description
- Many elements have **get**, **set**, **list** options which makes working with them intuitive
  - **firewall-cmd --list-services**
  - **firewall-cmd --get-services**
  - **firewall-cmd --add-service=service**
  - **firewall-cmd --remove-service=service**

Demo: Managing Ports and Services

- **firewall-cmd --list-all**
- **firewall-cmd --get-services**
- **firewall-cmd --add-service http**
- **cat /usr/lib/firewalld/services/http.xml**
- **firewall-cmd --add-port 123/tcp**
- **firewall-cmd --list-all**
- **firewall-cmd --runtime-to-permanent**

Demo: Managing Port Forwarding

- **firewall-cmd --add-forward-port=port=2022:proto=tcp:toport=22**
- from another host: **ssh -p 2022 <host-ip>**

5. Network Security

5.4 Using firewalld rich rules

Understanding Rich Rules

- Direct Rules allow admins to insert hand-coded rules
  - Direct rules are processed before anything in the zones
  - Not recommended to use them
- Rich rules use an expressive language to create custom rules that cannot be created with basic syntax
  - logging
  - port forwards
  - masquerading
  - rate limiting
  - man 5 firewalld.richlanguage

How to Compose Rich Rules

- Start with **firewall-cmd --add-rich-rule='<rule>'**
- Read **man 5 firewalld.richlanguage** for some rule examples

Demo: Composing Rich Rules

- firewall-cmd --permanent --zone=public --add-rich-rule='rule family=ipv4 source address=192.168.4.220 reject'
- firewall-cmd --add-rich-rule= 'rule service name=http log limit value=3/m accept'
- firewall-cmd --permanent --add-rich-rule='rule protocol value=igmp accept'
    - protocols match /etc/protocols
- firewall-cmd --permanent --add-rich-rule='rule family=ipv4 source address=10.0.0.0/24 port port=7900-7905 protocol=tcp accept’
- firewall-cmd --permanent --zone=public --add-rich-rule='rule service name="ssh" log prefix="ssh" level="notice" limit value="2/m" accept'

5. Network Security

5.5 Using UFW

Demo: Using UFW

- UFW is Uncomplicated Firewall, and was developed to work with an intuitive syntax
- **sudo ufw enable**
- **sudo ufw allow ssh (checks details in /etc/services)**
- **sudo ufw reject out ssh**
- **sudo ufw status**
- **sudo ufw delete reject out ssh**
- **sudo ufw deny proto tcp from 10.0.0.10 to any port 22**
- **sudo ufw reset**
- **sudo ufw app list**
- **sudo ufw app info Samba**
- **sudo ufw logging on**
- **man ufw**

5. Network Security

5.6 Configuring Firewalld NAT

Demo: Configuring NAT

- This demo adds server3 with 192.168.29.230 and 10.0.0.10 which is going to do NAT, and server4 with 10.0.0.11 which is behind NAT
- On server 3
  - **firewall-cmd --get-zones**
  - **firewall-cmd --zone=public --add-masquerade --permanent**
  - **firewall-cmd --zone=internal --change-interface=ens34**
  - **firewall-cmd --zone=public --add-interface=ens33**
  - **firewall-cmd --get-active-zones**
  - **firewall-cmd --list-all** (doesnt work)
  - **firewall-cmd --list-all --zone=internal**
  - **firewall-cmd ----zone=public --add-forward-port=port=2022:proto=tcp:toport=22:toaddr=10.0.0.11**
- On server1
  - **ssh student@192.168.29.230 -p 2022**

6. Logging and Auditing

6.1 Understanding Logging and Auditing

Understanding Logging and Auditing

- Logging is used to gather events that have been generated by different processes
- syslog is the legecy Linux log service, which later was replaced by rsyslog
- systemd-journald is the current logging service on systems that use systemd
- Auditing can be configured for more in-depth analysis of what is going on

6. Logging and Auditing

6.2 auditd basics

The audit system

- The audit system is a part of the kernel that allows for advanced tracking of security-related events
- Administrators define rules to determine what should be written to the audit log
- To use auditing, the **auditd** service must be started by systemd
- Messages are written to /var/log/audit/audit.log
- When auditd is not running, rsyslog receives kernel audit messages
- By default, only a few events are logged by auditd
- The **auditctl** command can be used for more advanced auditing configuration

Managing auditd

- The auditd process is managed through /etc/audit/auditd.conf
- This file contains paramaters that can be used to ensure that the audit log files are never getting to big
- Audit rules integrate deeply into the kernel, which makes **systemctl restart auditd** inadequate for updating rules. Instead, use **augenrules --load** or restart your system

Reading audit logs

- The /etc/audit.d/audit.log file contains logged messages
- Differs tools exist to make filtering a bit easier
- Alternatively, feel free to use **less** or any other pager to browse audit.log contents
- Notice the auid in the audit log, this is the audit UID, which is the UID of the original user that logged in (before su or sudo)

Creating custom rules

- file system rules (AKA watches) can be used to audit access to files and directories
- system call rules audit execution of system calls
  - Notice that a watch rule can also be written as a system call rule
- control rules configure the audit system itself
- auditd applies the first rule that matches, so if you have a generic rule for /etc and a specific rule for /etc/shadow, make sure to list the /etc/shadow rule first!

Using watch rules

- The basic syntax of the watch rule is **auditctl -w file -p permissions -k key**
- **-w** specifies the file or directory to watch
- **-p** is for permissions
  - **r** is read access
  - **w** is write access
  - **x** is execute access
  - **a** watches for changes to file attributes
- **-k** is used to set a key that makes it easy to find a specific audit record
- Watch rules do not cross filesystem boundaries: subdirectories are included, but not if they are a separate mount

Demo: using watch rules

- **auditctl -w /etc/passwd -p wa -k passwd-access**
- **cat /etc/passwd**
- **grep passwd-access /var/log/audit/audit.log**
- **auditctl -w /bin -p x**
- **tail /var/log/audit/audit.log**

Auditing system calls

- system calls are instructions used by applications to get access to kernel functionality
- The audit subsystem can trace each system call
  - Use **-S** to specify which system call is involved
  - Use **-F** for advanced filtering options
  - Use **-C** for field comparison (like auid=1000)
- When auditing system calls, the **-a exit** option is normally used, to ensure the auditing happens at exit of the system call
- Commonly **-a always,exit** is used, which will always audit the rule. To never audit an event, use **-a never,exit**
  - **auditctl -a never,exit -F path=/usr/sbin/crond -F perm=x**

Auditing system calls

- When auditing system calls, on 64-bit systems use **-F arch=b64** as well as **-F arch=b32**, because even on 64-bits systems 32-bits system calls may be used
- Add **-C auid!=4294967295** to exclude system calls made by auditing
- Auditing system calls creates a lot of overhead!

Understanding audit UIDs

- uid: the observerd UID. If user anna with UID 1002 has opened a root shell using **su**, the uid will show as 0. This also shows as the effective UID (euid)
- auid: the audit UID, which is the UID of the original user that has logged in. In the previous example, that would be the UID of user anna
- obj_uid: the object UID is the UID assigned to an object. This is about file owners and will show the UID of the user owner of a file

Examples: Auditing system calls

- **auditctl -a exit,always -F dir=/home/ -F uid=0 -C auid!=obj_uid** will audit all file access in the /home directory by uid=0 (root) where the audit UID is not the same as the object UID (which is the owner of the file)
- **auditctl -a exit,always -F arch=b32 -S unlink -S unlinkat -S rename -S renameat -k DELETE**
- **auditctl -a exit,always -F arch=b64 -S unlink -S unlinkat -S rename -S renameat -k DELETE**
  - These two rules will audit all file deletion. Notice that on 64-bits platforms you need a rule for 64-bits as well as 32 bits system calls
- Use **grep DELETE /var/log/audit/audit.log** to monitor

Managing Rules

- **auditctl -l** will list current effective rules
- **auditctl -D** removes all rules
- To make rules persistent, add them to /etc/audit/rules.d/audit.rules
- Do NOT edit /etc/audit/audit.rules on Red Hat, it will be overwritten at restart
- After adding audit rules, use **augenrules --load** to activate them

Reading Audit Logs

- You can directly read messages in /var/log/audit/audit.log
- **ausearch** makes interpreting logged information a bit easier
- **ausearch -p 1234 --raw** shows raw messages logged by PID 1234
- **ausearch -a 2233** shows messages with audit event ID 2233
- **ausearch -f /etc/passwd** shows all events related to a specific file
- **ausearch -m AVC** shows all events with a specific message type
- **ausearch -k mykey** shows all events with a specific key
- **autrace /bin/ls** generates an audit trace displaying all system calls

EX415: Prepackaged Audit Rule Sets

- RHEL comes with sets of sample rules
- You'll find these in /usr/share/audit/sample-rules/
- To use them, copy to /etc/audit/rules.d/ and use **augenrules --load** to reload
- While doing so, read the sample rules to see if there are any dependencies

7. SELinux and AppArmor

7.1 Understanding Mandatory Access Control

Managing Linux Access Control

- Historically, Linux has never been developed with security in mind
- Different security solutions with a focus on different aspects of Linux exist
- As a result, it's easy to overlook specific parts of Linux security
- In the 1990's solutions for Mandatory Access Control have been introduced
- The main solutions are SELinux and AppArmor

SELinux versus AppArmor

SELinux	AppArmor
Locks down everything; if it isn’t allowed, it’s denied	Works with profiles to secure specific services
More complex to learn	Relatively easy to learn
Offers more advanced features, such as multi-level security	
Uses filesystem labels	
Default in Red Hat	Default in Ubuntu

7: SELinux and AppArmor

7.2 Confining Services with AppArmor

Managing AppArmor

- Enable AppArmor through its systemd service
- Ensure an application profile is available in /etc/apparmor.d
- Alternatively, create a profile yourself, using:
  - **aa-genprof /your/application**
  - Run the application from another terminal
  - Press **s** to scan for application events
- Loaded profiles are in /sys/kernel/security/apparmor
  - Use **aa-status** as an alternative

Tip: AppArmor CLI tools all start with **aa-**

- Install all AppArmor tools

Demo: Using AppArmor

- **sudo systemctl status apparmor**
- **sudo apt search apparmor**
- **sudo apt install apparmor-utils apparmor-profiles apparmor-profiles-extra**
- **sudo aa-status**
- **sudo aa-genprof /usr/bin/vim**
- Scan for events and finish profile
- **vim /etc/motd**
- **vim /etc/apparmor.d/usr.bin.vim.basic**
- **nano /etc/apparmor.d/usr.bin.vim.basic**
- **rm /etc/apparmor.d/usr.bin.vim.basic**

7: SELinux and AppArmor

7.3 Understanding SELinux

Understanding SELinux

- SELinux provides a policy that identifies which source object has access to which target object
- The rules in the policy are based on labels
- Labels consist of three parts:
  - User
  - Role
  - Type
- For base SELinux use, the context type is used
- When a specific source context type requests access to a specific target context type, SELinux checks if there is a rule that allows this

Managing SELinux Modes and States

- SELinux is either disabled or enabled
- This is a state in the Linux kernel, and can only be set while booting: **selinux=0** or **selinux=1**
- If SELinux is enabled, you can toggle between **permissive** and **enforcing** mode
- In **permissive** mode, nothing is blocked, but all SELinux events are logged
- In **enforcing** mode SELinux is fully operational
- Change /etc/sysconfig/selinux contents to set persistent state
- Use **getenforce** to get the current mode
- Use **setenfoce** to set the current mode

7: SELinux and AppArmor

7.4 Applying Labels to Manage SELinux File Access

Using Labels

- All items are using context labels
- Use the **-Z** option to print label information
- The context type is the most important part of the labels
- Use **semanage fcontext** to set context labels on files, this will write to the policy
- After setting context labels on files, use **restorecon** to apply from policy to the inodes
- Use **semanage port** to set context labels on network ports

Demo: Configuring File Context

- dnf install httpd -y
- mkdir /web
- echo hello world > /web/index.html
- sed -i -e 's/^DocumentRoot.*/DocumentRoot \/web/' /etc/httpd/conf/httpd.conf
- sed -i -e 's/\/var\/www/\/web/' /etc/httpd/conf/httpd.conf
- systemctl enable --now httpd
- curl localhost
- setenforce 0
- curl localhost

Demo: Configuring File Context

- **grep AVC /var/log/audit/audit.log**
- **setenforce 1**
- **ls -Z /var/www/**
- **ls -Zd /web**
- **man semanage-fcontext**
- **semanage fcontext -a -t httpd_sys_content_t "/web(/.*)?"**
- **restorecon -Rv /web**
- **curl localhost**

7: SELinux and AppArmor

7.5 Applying Labels to Manage SELinux Port Access

Demo: Managing SELinux Port Access

- **sed -i -e 's/^Listen.*/Listen 82/' /etc/httpd/conf/httpd.conf**
- **systemctl restart httpd**
- **systemctl status httpd**
- **grep AVC /var/log/audit/audit.log**
- **semanage port -a -t http_port_t -p tcp 82**
- **systemctl restart httpd**
- **systemctl status httpd**

7: SELinux and AppArmor

7.6 Configuring Booleans

Understanding Booleans

- A boolean is an on/off switch that allows you to easily apply settings in SELinux
- Booleans are used in addition to context labels
- Use **getsebool -a** to get a list of all booleans
- Use **setsebool -P** to make persistent changes to booleans

7: SELinux and AppArmor

7.7 Troubleshooting SELinux Access

Troubleshooting SELinux

- SELinux events are logged to the audit log in /var/log/audit/audit.log
- Messages in the audit log are not always very readable
- Apart from that, there is **sealert**, which is part of the setroubleshoot-server package and logs readable messages to your logging system
- Look for the **sealert** command example in your log files and run this command to find out what to do

7: SELinux and AppArmor

7.7 SELinux Users

EX415: SELinux Users

- SELinux users are profiles that are added to Linux users that log in
- Use **semanage user -l** to list them
- SELinux users are mapped to Linux users; use **semanage login -l** to show these mappings
- To map an existing Linux user to an SELinux user, use **semanage login -a -s sysadm_u username**
- To modify the default mapping, use **semanage login -m -s user_u -r s0 __default__**
- Use **id -Z** to check current SELinux user associations

EX415: Understanding SELinux Users

- user_u: used for standard users. No access to sudo or su, no SUID programs
- sysadm_u: does have access to sudo and su (which need further configuration), no ssh access unless the ssh_sysadm_login boolean is set
- staff_u: can use sudo, but not su.

8: fapolicyd

Understanding fapolicyd

- fapolicyd is an optional service that can be used to deny unknown applications
- In fapolicy, applications are only allowed if they are trusted
- Applications can be known if they match the RPM origins: if a package is installed through **yum** or **dnf** it is automatically trusted
- To add custom file, use **fapolicyd-cli --file add**, followed by **fapolicyd-cli --update**

Using fapolicyd

- **dnf install fapolicyd**
- **systemctl enable --now fapolicyd**
- **cp /bin/ls /tmp/ls**
- as non-root user: **/tmp/ls** will be denied
- **fapolicyd-cli --file add /tmp/ls --trust-file myapp**
- **fapolicyd-cli --update**
- as non-root user: **/tmp/ls** will now work

9: USBGuard

EX415: USBGuard

- USBGuard uses a daemon process that works with rules which are stored in a configuration file to allow or deny access to USB devices
- The workflow is as follows
  - Start and enable the daemon
  - Generate a policy that allows devices that were found
  - Plug a new device, and use **usbguard list-devices --blocked** to see it
  - Add it persistently using **usbguard allow-devices -p 5** to allow persistent access

EX415: Configuring USBGuard

- **usbguard generate-policy > /etc/usbguard/rules.conf** generates a configuration that allows all currently connected devices, all else is blocked
- **usbguard generate-policy -X** will do the same but without hashes (which are hard to handle anyway)
- **systemctl enable --now usbguard**
- **usbguard list-rules** shows current rules
- **usbguard list-devices** shows current devices
- **usbguard allow-device 2** allows the device in rule #2 from the list-devices output non-persistently
- **usbguard allow-device -p 2** allows device #2 persistently
- **usbguard list-devices --blocked** list devices that are connected, but blocked

EX415: Configuring the Daemon

- /etc/usbguard/usbguard-daemon.conf has default settings
  - ImplicitPolicyTarget=block will block unknown devices
  - IPCAllowedUsers=usernames allows users to access the daemon
  - IPCAllowedGroups=groupnames: same for groups
- **usbguard add-user -u luser --devices=list**

10: OpenSCAP

SCAP

- SCAP is Security Compliance Automation Protocol
- OpenSCAP offers tools for implementing and enforcing security policies
- It also comes with some standard policies, provided in the SCAP Security Guide
- Using OpenSCAP allows for making a complete security audit, as well as generating a solution for remediation
- SCAP Workbench is the graphical tool that can be used to work with OpenSCAP
    - It can scan a local machine, or use its internal SSH to manage remote machines
- **oscap** is the command line utility
- Both work with predefined content which is in /usr/share/xml/scap/sgg/content

Security Guide

- The SCAP Security Guide has different policies to match platforms
- Within a policy you'll find profiles that are developed to deal with specific situations

Using the **oscap** tool

- The **oscap** tool works directly on XML files in /usr/share/xml/scap/sgg/content/
- Files with names that end in -ds.xml are XCCDF data stream files
- Use **oscap info** on any of these files to find available profiles
- To scan a system, install **openscap-scanner** and **scap-security-guide** and select the XCCDF file and profile you want to use
- Next, use the **oscap xccdf eval** command with the appropriate options to scan the local system. While doing so, specify the profile to use, which you've found using **oscap info**:
  - **oscap xccdf eval --profile xccdf_org.ssgproject.content_profile_stig --results /root/results.xml /usr/share/xml/scap/content/ssg-rhel9-ds.xml**

Using the **oscap** tool

- The result of the scan by default is in XML format. To convert it to a readable format, use **oscap xccdf generate report results.xml > results.html**

Customizing OpenSCAP Policy

- In most situations, only customized profiles in the security guide are used
- To do so, you use an existing profile that you adjust by selecting and removing rules
- The new profile is saved into a tailoring file, which can next be used on all the systems you want to scan
- To scan another system with the custom profile, use **oscap xccdf eval --profile *custom-profile-id* --tailoring-file *tailoring-file.xml* --results /root/results.xml /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml**

EX415: Using OpenScap for Remediation

- On all machines: **sudo dnf install scap-workbench scap-security-guide openscap**
- On the workstation using **scap-workbench**: create the customization file and save as ~/whatever.xml
- **scp whatever.xml targetserver:**
- On target server, from a root shell: **oscap xccdf eval --profile xccdf_name_that_you_provided --tailoring-file /home/student/whatever.xml --results /root/lab-results.xml /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml**
  - Use **oscap info whatever.xml** to find the xccdf name of the profile
  - Tailoring file MUST be an absolute filename!
  - Use [Tab][Tab] on **oscap xccdf eval** for options

EX415: Using OpenScap for Remediation

- On target server, convert the results to HTML: **oscap xccdf generate report lab-results.xml > lab-results.html**
  - Do NOT use the **--output-file** option, it doesn't work
- Copy the files to the workstation machine
- On the workstation as non-root, generate the playbook: **oscap xccdf generate fix --profile xccdf_name_that_you_provided --tailoring-file whatever.xml --fix-type ansible --result-id "" lab-results.xml > fix.yml**
- Add become: true to fix.yml
- **echo targethostname > inventory**
- Run **ansible-playbook -K -i inventory fix.yml**

NTS: EX415: hard-to-find Options

- Work with sections, disable sections you don't need
- Tip! Hide all section subtrees so that you only see the main section headers
- Look for **Services > FTP**; **Services > Telnet**
- Look for **System Settings > Network Configuration and Firewalls > Firewalld**