Istituto Nazionale di Fisica Nucleare

Corso RedHat per sistemisti INFN

RHEL/SL/CentOS 7

Il processo di avvio

- All'accensione il BIOS/UEFI esegue un Power On Self Test (POST) ed inizializza alcuni device
- Il BIOS/UEFI esegue il codice necessario all'avvio (boot loader) leggendolo da uno dei boot device disponibili.
- Il boot loader carica in memoria ed esegue un kernel presente sul boot device caricando in memoria anche un initial ram disk (initramfs).
  - L'initramfs contiene tutti i driver necessari ad eseguire systemd.

Il processo di avvio

- Il kernel decomprime l'initramfs, carica i driver necessari ed esegue il binario systemd presente in memoria.
- Il processo systemd esegue tutte le unita' del target initrd.target montando il filesystem di sistema in /sysroot.
- Avviene uno scambio tra al root directory del ramdisk ed la directory /sysroot, che diviene la nuova “/” (pivot)
- Systemd carica se stesso utilizzando la copia presente su disco rigido, ed esegue tutte le unita' del target di default, fino a giungere al prompt di login

UEFI

- CentOS7 ha migliorato la compatibilita' con UEFI e SecureBoot
- Limiti del BIOS
  - BIOS usa MBR -> Max 4 partizioni da 2TB l'una
  - BIOS lavora a 16 bit -> 1MB dimensione massima dell'esegiubile
- UEFI usa GPT al posto di MBR
  - Numero di partizioni virtualmente infinito
  - Massima dimensioni del disco ~9ZB
  - Contiene un MBR per retrocompatibilita'
- UEFI lavora a 64bit
  - Eseguibile del firmware di dimensioni maggiori
  - Indirizza piu' memoria -> carica in RAM piu' velocemente gli eseguibili di bootstrap
- Coreboot/TinyCore: una implementazione opensource delle specifiche UEFI
  - Presente come virtual firmware in Qemu/KVM

Boot UEFI

- UEFI carica gli eseguibili da una partizione FAT32 taggata come ESP (EFI System Partition)
  - Gli eseguibili si trovano di norma in /EFI (es.: /EFI/RedHat/grubx64.efi)
- Puo' caricare sia bootloader (GRUB2, rEFInd) che direttamente un kernel
  - Il parametro relativo al root file system deve essere aggiunto nella compilazione del kernel
  - Il UEFI BootManager determina quale dei vari bootloader presenti nella partizione ESP debba essere eseguito
    - Puo' essere configurato dal sistema in esecuzione con il comando `efibootmgr`

SecureBoot (the bad)

- Funzione di UEFI che consente di eseguire al boot solo bootloader o kernel firmati con un meccanismo di chiavi privata/pubblica
- CentOS7 e' la prima versione CentOS ad essere totalmente compatibile con SecureBoot
- Un computer certificato Windows8 o Windows10 deve avere UEFI SecureBoot abilitato e contenere le chiavi pubbliche di Microsoft
  - Per Windows10 non e' obbligatorio che sia disabilitabile
  - Di fatto le chiavi Microsoft sono le uniche ad essere incluse
- E' possibile, pagando 99$ far firmare a Microsoft un bootloader o un kernel
  - Ma... Microsoft non firma software rilasciato con licenza GPLv3!!

SecureBoot (the good)

- RedHat e TheLinuxFoundation hanno "pagato il pizzo" e fatto firmare due diversi bootloader Shim e PreLoader
  - Si inseriscono tra UEFI e Grub2
  - In CentOS7 viene usato Shim
- CentOS7 estende la "chain of trust" a Grub2 ed al Kernel
  - Il kernel viene compilato con il parametro CONFIG_MODULE_SIG a true
  - Non possono venire montati moduli del kernel non firmati (utile contro i rootkit!!!)
  - Shim introduce la sua infrastruttura di chiavi (MOK): Grub2 ed il kernel non devono essere firmati da Microsoft

Istituto Nazionale di Fisica Nucleare

GRUB2

GRUB2 vs. GRUB legacy

- GRUB2 e' piu' flessibile:
  - ~100 comandi e una shell avanzata
  - Il file di configurazione e' uno script
- Supporta piu' filesystem (es. NTFS, HFS+, ZFS)
- Legge direttamente da RAID e LVM
- Ha una architettura piu' modulare:
  - Decine di moduli caricabili all'avvio

Struttura di GRUB2

- Il **kernel** svolge le funzioni fondamentali:
  - L'allocazione della memoria, l'interprete dei comandi, il loader dei moduli ed una shell minimale
- I **moduli** aggiungono funzionalita' al kernel e possono essere caricati o inclusi in una immagine core. Sono esempi di moduli:
  - Supporto ai vari filesystem, comandi aggiuntivi, driver di periferiche (porta seriale, interfacce di rete)
- L'immagine di **core** e' composta dal kernel, da un esiguo numero di moduli fondamentali e dalla *prefix_string* (percorso contenente i moduli aggiuntivi)
  - Esempio di prefix_string: (hd0,msdos1)/boot/grub
  - La struttura della core image dipende dalla piattaforma
- In acluni casi e' presente una immagine di **boot**, ad esempio quella che viene scritta nell'MBR del BIOS

Boot di GRUB2 (BIOS vs. UEFI)

BIOS

MBR GRUB IMAGE

KERNEL

UEFI

SHIM IMAGE su EFS

GRUB IMAGE su EFS

KERNEL

Installare GRUB2

- GRUB2 viene installato insieme al sistema operativo.
- Puo' essere reinstallato successivamente per:
  - Sovrascrivere una installazione precedente (ad esempio di un'alltro sistema operativo sullo stesso disco)
  - Riparare una eventuale corruzione/mancanza di dati
  - Trasferire il sistema di boot su un altro device
- Il comando di installazione e':
  - `grub2-install <device> #BIOS`
  - `yum reinstall grub2-efi shim #UEFI`
- Crea un'immagine *core* contenente il kernel ed i moduli necessari e successivamente:
  - Nell'architettura BIOS scrive l'immagine di *boot* nel MBR
  - Nell'architettura UEFI copia l'immagine *core* nella partizione ESP

Configurare GRUB2

- La configurazione di GRUB2 si trova nei file:
  - `/boot/grub2/grub.cfg #BIOS`
  - `/boot/efi/EFI/centos/grub.cfg #UEFI`
- E' piu simile ad un bash script che ad un file di configurazione tradizionale
  - Viene creata all'installazione del bootloader o tramite il comando `grub2-mkconfig`
  - Il comando crea la configurazione a partire dagli script presenti in `/etc/grub.d` e dal file di personalizzazione `/etc/default/grub`
- Ad ogni update del kernel il file viene modificato dagli script di post-installazione del pacchetto rpm
- Il file di configurazione non deve venire alterato direttamente. Esistono due meccanismi per farlo: **grubby** e **grub2-mkconfig**
- Modifiche temporanee possono essere introdotte all'avvio del sistema, quando viene mostrato il menu di selezione

Grub.cfg: il menuentry

- La sezione *menuentry* del file di configurazione grub.cfg rappresenta la singola voce di menu selezionabile all'avvio

```
menuentry 'CentOS 7' --class centos --class gnu-
   linux --class gnu --class os --unrestricted
   $menuentry_id_option 'centos-7-test' {
        insmod gzio
        insmod part_msdos
        insmod ext2
        set root='hd0,msdos1'
        linux16 /boot/vmlinuz.el7.x86_64
   root=/dev/sda1 ro
        initrd16 /boot/initramfs.img
}
```

Modifiche temporanee all'avvio

- All'avvio del sistema, quando Grub mostra il menu di selezione, evidenziare una voce e premere il tasto **e**
- Con il cursore raggiungere la linea da modificare e procedere
  - per aggiungere o eliminare un parametro del kernel posizionarsi sulla linea che inizia con **linux16** (per sistemi BIOS) o **linuxefi** (per sistemi UEFI)
- Premere `Ctrl-x` per avviare o `ESC` per ritornare al menu
- All'avvio il tasto **c** al posto di **e** porta alla grub-shell

Configurare GRUB2: grubby

- L'utility *grubby* ci permette di modificare la configurazione di grub da linea di comando
  - Mette al riparo da errori di sintassi
  - Al contrario di *grub2-mkconfig* ogni modifica precedente e' preservata
- Modifica direttamente i file `grub.cfg` e `grubenv`, che contiene alcune variabili utilizzate in `grub.cfg`
- Le opzioni di configurazione sono limitate alla modifica del kernel di default, degli argomenti relativi e all'inserimento di nuovi kernel

Grubby: esempi di esecuzione

```
~]# grubby --default-kernel
/boot/vmlinuz-3.10.0-693.el7.x86_64

~]# grubby --set-default \
 /boot/vmlinuz-3.10.0-229.4.2.el7.x86_64

~]$ grubby --info=ALL
index=0
kernel=/boot/vmlinuz-3.10.0-693.el7.x86_64
……

~]# grubby --remove-args="rhgb quiet" \
 --args=console=ttyS0,115200 --update-kernel \
 /boot/vmlinuz-3.10.0-229.4.2.el7.x86_64
```

Configurare GRUB2: grub2-mkconfig

- E' necessario creare da zero il file grub.cfg quando si vuole agire a basso livello sulla configurazione del boot loader
  - Aggiungere voci al menu (oltre a quelle relative ai kernel installati)
  - Cambiare l'ordine delle voci
  - Ripristinare la configurazione a seguito di un problema
- Il comando `grub2-mkconfig` permette di
  - Creare la configurazione automaticamente individuando i sistemi operativi ed i kernel installati
  - Personalizzare in parte o totalmente il menu
- Esegue in ordine alfabetico gli script contenuti in /etc/grub.d

Configurare GRUB2:grub2-mkconfig

- **00_header**: carica i default settings da /etc/default/grub
- **01_users** legge l'eventuale password da user.cfg
- **10_linux** individua i kernel nella partizione di sistema di CentOS.
- **30_os-prober** individua eventuali altri sistemi operativi installati in altri device o partizioni
- **40_custom** un template che puo' essere utilizzato per aggiungere personalizzazioni

Configurazione semplice

- Il file /etc/default/grub contiene i parametri che consentono di modificare le voci di menu che vengono create automaticamente.
- I possibili parametri da includere nel file sono disponibili sulla documentazione GNU
  - https://www.gnu.org/software/grub/manual/grub/grub.html#Simple-configuration
- Non e' possibile aggiungere voci di menu

Configurazione semplice

```
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto rhgb quiet"
GRUB_DISABLE_RECOVERY="true"
```

- Se il valore di timeout viene impostato a zero, non verra' visualizzato il menu al boot della macchina
- La pressione di qualsiasi tasto prima dell'esecuzione del boot loader causera' l'interruzione del boot e la visualizzazione del menu

Configurazione semplice

```
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto rhgb quiet"
GRUB_DISABLE_RECOVERY="true"
```

- GRUB_DEFAULT puo' assumere il valore dell'**id** di una sezione *menuentry* sul file di configurazione
- La keyword **saved** indica che, come default, verra' utilizzato:
  - L'ultimo kernel selezionato all'avvio se il parametro GRUB_SAVEDEFAULT e' uguale a "true"
  - Il valore specificato nella variabile **saved_entry** del file `/boot/grub2/grubenv` (impostabile attraverso il comando grub2-set-default

Configurazione semplice

```
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto rhgb quiet"
GRUB_DISABLE_RECOVERY="true"
```

- Di default `grub2-mkconfig` inserisce i kernel piu' datati in un sotto-menu (Ubuntu style)

Configurazione semplice

```
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto rhgb quiet"
GRUB_DISABLE_RECOVERY="true"
```

- Grub2 puo' acquisire input e dirigere output da diversi device
- La configurazione per la porta seriale potrebbe essere:

```
GRUB_TERMINAL="serial"  #Vale per INPUT e OUTPUT
GRUB_SERIAL_COMMAND="serial --speed=9600 --unit=0 --word=8 --
parity=no --stop=1"
```

- Puo' perfino utilizzare la scheda audio come modem!!!

Configurazione semplice

```
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto rhgb quiet"
GRUB_DISABLE_RECOVERY="true"
```

- Disabilita la creazione automatica di una voce di menu per partire in "single mode"

Configurazione Avanzata

- E' fortemente sconsigliato agire direttamente sul file di configurazione
- Una voce di menu non inclusa tra quelle individuate dal configuratore puo' essere aggiunta nel file `/etc/grub.d/40_custom`
- Chi volesse sostituire completamente il menu di grub con uno creato ad-hoc deve:
  - Eliminare tutti i file presenti in `/etc/grub.d/` tranne **`00_header`** e **`40_custom`**
  - Inserire il proprio menu nel file `40_custom`
- Attenzione: l'installazione di un nuovo kernel modifica il file di configurazione. Per evitarlo e' necessario inserire nel file `/etc/sysconfig/kernel` il parametro **`UPDATEDEFAULT=no`**

Bootloader password

- E' possibile proteggere con password una o piu' voci nel menu
- Di default l'accesso ad un entry e' protetto solo per quanto riguarda le modifiche (attuate all'avvio della macchina). Per proteggere anche la selezione e' necessario eliminare il parametro `--unrestricted` dalle sezioni *menuentry* del file di configurazione
- Il comando `grub2-setpassword` scrive la hash della password nel file `/boot/grub2/user.cfg`
- L'utente e' definito in `/etc/grub.d/01_users`, il default e' **root**

Istituto Nazionale di Fisica Nucleare

Procedure di avvio di emergenza

Rescue mode

- Il rescue mode e' un target systemd che consente di avviare il sistema in single user per poter modificare un'installazione che ha problemi nella fase di avvio dei servizi non fondamentali
- Il sistema prova a montare tutti i filesystem locali configurati ma non attiva interfacce di rete
- E' equivalente al "single mode" della CentOS 6 ma richiede l'inserimento della password di root
- Il parametro da aggiungere alla linea **linux16** o **linuxefi** del menu grub e'
  - `systemd.unit=rescue.target`

Emergency mode

- L'emergency e' un target systemd che consente di avviare il sistema che fornisce l'ambiente con il minimo delle funzionalita' possibili
- Viene montato il solo root filesystem in modalita' read-only. Le interfacce di rete sono disattivate
- Richiede l'inserimento della password di root
- Il parametro da aggiungere alla linea **linux16** o **linuxefi** del menu grub e'
  - `systemd.unit=emergency.target`

Shell di debug

- Systemd ha la possibilita' di eseguire una shell all'avvio del sistema, prima degli altri servizi, in modo da controllare il processo di avvio e scoprire la causa di eventuali problemi
- Il parametro da aggiungere alla linea **linux16** o **linuxefi** del menu grub e'
  - `Systemd.debug-shell`
- Puo' essere utile attivare il debug di systemd aggiungendo
  - `systemd.log_level=debug`
- La combinazione di tasti `Ctrl-Alt-F9` seleziona la console connessa alla shell. Nessuna autenticazione e' richiesta

Reset della password di root

- Il reset della password di root non puo' avvenire ne' in emergency mode ne' in rescue mode
- Il metodo piu' veloce e' quello di abilitare la debug-shell e modificare la password
- Se la debug-shell non fosse disponibile:
  - Aggiungere al boot il parametro `rd.brake` ed `enforcing=0` (in caso abbiate a che fare con SELinux)
  - L'avvio si blocca prima del pivot, eseguendo la shell dell'initramfs con il root filesystem montato read-only in `/sysroot`
  - Rimontare rw `/sysroot`, eseguire un `chroot` su `/sysroot` e modificare la password di root
  - Per compatibilita' con SELinux, eseguire un `restorecon` di `/etc/shadow`

Istituto Nazionale di Fisica Nucleare

Initramfs e parametri di avvio del kernel

Dracut

- `dracut` e' il nome del sistema di creazione del ramdisk di avvio ed il comando per crearlo
- Includere *udev* nell'initramfs
  - Un initramfs generico che si adatta semplicemente a diverse configurazioni
- Permette di avviare sistemi direttamente da RAID (md e dm), LVM, DM-Crypto (LUKS anche con chiave esterna), NFS, CIFS, iSCSI

Dracut

- Il comando per creare un initramfs e' semplicemente `dracut`
  - Senza l'opzione `-f` si rifiuta di sovrascrivere un ramdisk esistente
- Verranno inclusi tutti i moduli del kernel ed le configuarazioni relativi al sistemi in esecuzione
- Il file creato (/boot/initramfs-$(uname -r).img) e' un archivio CPIO compresso
- Per spacchettare il contenuto

```
mkdir /tmp/initramfs/; cpio -
I /boot/initramfs-$(uname -r).img
```

- Per mostrare la lista o il contenuto dei file inclusi eseguire il comando `lsinitrd [-f file]`

Dracut

- Il sistema di creazione del ramfs e' composto da moduli che possono essere incusi o meno nell'immagine
- Lista dei moduli

```
dracut --list-modules
```

- Aggiungere moduli al ramfs

```
dracut --add "ifcfg nfs" initramfs.img
```

- Eliminare moduli dal ramfs

```
dracut -o "lvm" initramfs.img
```

Parametri di avvio

- Al menu di grub e' possibile modificare il comportamento di Dracut per l'avvio in corso, aggiungendo parametri alla linea relativa al kernel
- La lista dei possibili parametri di avvio relativi all'initramfs e' consultabile con il comano `man dracut.cmdline`
- Altri parametri, che sono interpretati direttamente dal kernel, sono elencati su https://www.kernel.org/doc/Documentation/admin-guide/kernel-parameters.txt

Parametri di avvio

- Sono raggruppabili in categorie:
  - Generici standard: root=<path>, ro, rw, rootfallback=<action>, rd.auto, resume...
  - Generici avanzati: rd.driver.<blacklist|pre|post>, rd.retry...
  - Debug: rd.shell, rd.debug, rd.break, rd.udev.debug....
  - Localizzazione: keymap, fonts, unicode...
  - Storage: LVM, RAID, NFS, CIFS, iSCSI, DM-Crypto...
  - Network: ip, ifname, biosdevname, vlan, bond, team, bridge....
  - Live-system: squash-fs, loop device images...

Esempi

- Avvio da UUID o LABEL

```
root=LABEL=Root
root=/dev/disk/by-uuid/3f5ad593-4546-4a94-a374-bcfb68aa11f7
root=UUID=3f5ad593-4546-4a94-a374-bcfb68aa11f7
```

- Disabilitare il renaming delle interfacce di rete

```
biosdevname=0
```

- Avvio da NFS con attivazione della rete via dhcp sull'interfaccia *eno1*

```
ip=eno1:dhcp root=nfs:nfsserver:/root_path:
```

- Configurare la tastiera italiana

```
rd.vconsole.font=latarcyrheb-sun16
  rd.vconsole.keymap=it rd.locale.LANG=it_IT.UTF-8
```

Troubleshooting

- Se l'avvio si interrompe a causa di errori durante l'esecuzione dei comandi dell'initramfs:
  - Eliminare i parametri "rhgb" e "quiet"
  - Aggiungere rd.shell (presenta una shell interattiva di emergenza)
  - Aggiungere rd.debug (equivalente ad un -x nell'esecuzione dei comandi bash)
  - Eseguire l'avvio e, una volta giunti alla shell, controllare il contenuto di `/run/initramfs/rdsosreport.txt`
  - Eventualmente utilizzare rd.brake per interrompere l'avvio a diversi stadi dell'esecuzione in ramfs

Personalizzazione del ramfs

- Per includere piu' file o directory, alla creazione del ramfs, e' necessario utilizzare il parametro `--include` seguito dal sottoalbero completo che deve essere aggiunto
- Uno o piu' comandi possono essere inseriti, insieme alle librerie da cui dipendono, attraverso il parametro `--install`

```
dracut --install 'strace ssh' initramdbg.img
```

Esempio

```
# mkdir -p rd.to.add/etc/cmdline.d
# mkdir -p rd.to.add/etc/conf.d
# echo "ip=dhcp" >>
rd.to.add/etc/cmdline.d/mycmdline.conf
# echo export FOO=testtest >>
rd.to.add/etc/conf.d/testvar.conf
# dracut --include rd.to.add newinitram.img
```

I moduli di Dracut

- Dracut e' composto da moduli che vengono eseguiti in vari stadi **(hook)** dell'esecuzione dell'initramfs
  - Gli hook sono: cmdline, pre-udev, pre-trigger, pre-mount, mount, pre-pivot, cleanup
  - Pagina di manuale `dracut.modules(7)`
- I moduli si trovano nella directory `/usr/lib/dracut/modules.d/`
- Il comando dracut `--list-modules` mostra I moduli disponibili

Creare un modulo

- Creare un modulo e' molto semplice:
  - Creare una directory in /usr/lib/dracut/modules.d/
  - Creare nella directory uno script modules_init.sh che definisce l'hook relativo al modulo e i comandi da eseguire quando l'hook viene lanciato
- Per includere il modulo appena creato in un initramfs e' sufficiente eseguire
  - `dracut --add newmodule -f \ /boot/initramfs_custom.img`