Universidad Nacional Autónoma de México
Facultad de Ciencias

Administración de Sistemas UNIX/Linux

Semestre: 2025-2

Práctica 5

`Compilando el Kernel vol.2`

Profesor: Luis Enrique Serrano Gutiérrez
Ayudante: Erick Iram García Velasco

Equipo:
Bonilla Reyes Dafne 319089660
Cabrera Ramirez Carlos 316699910
García Ponce José Camilo 319210536
García Sánchez Abigail 423026522

Abril, 2024

Objetivos

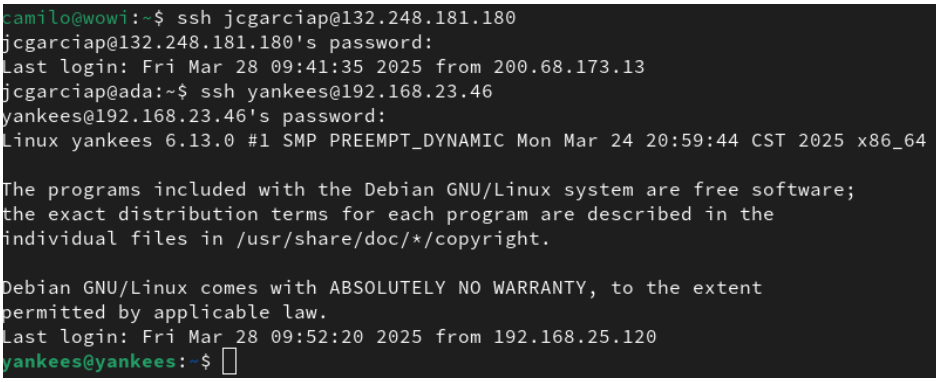
- Familiarizarnos con más opciones de configuración disponibles.
- Conocer más comandos para la configuración del kernel.

Desarrollo

Conexión a la computadora

1. Para hacer la conexión a la computadora del laboratorio primero nos conectamos usando **SSH** al sistema Ada, esto lo hacemos usando nuestro usuario y contraseña. Después, dentro de Ada nos vamos a conectar a la computadora. Todo esto lo hacemos usando los siguientes comandos:

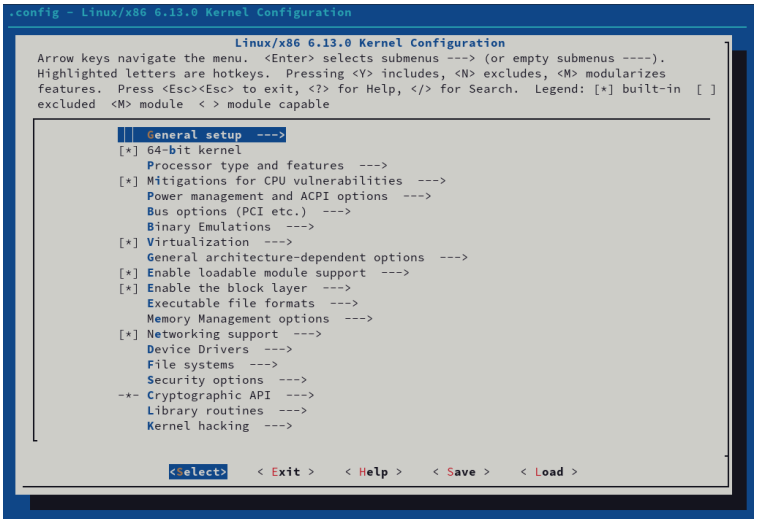
```
ssh jcgarciap@132.248.181.180
ssh yankees@192.168.23.46
```



Asignación y eliminación de módulos

1. Para revisar la configuración de los módulos del kernel que realizamos en la práctica pasada, usamos el siguiente comando

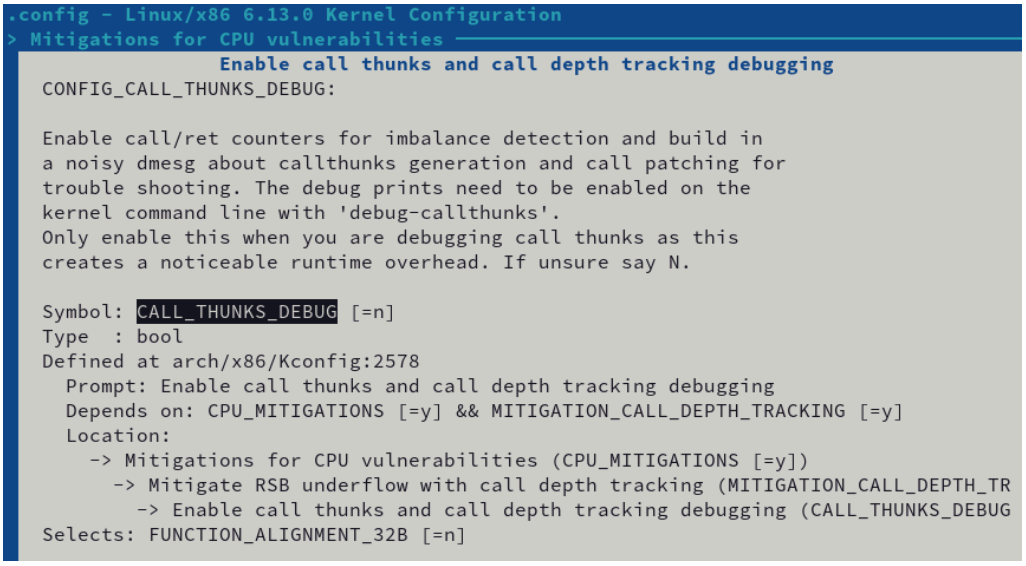
```
sudo makefile config
```



2. Una vez dentro de las configuraciones del kernel, revisamos que módulos vamos a agregar y cuáles vamos a eliminar:

SECCIÓN	MÓDULOS A ELIMINAR	MÓDULOS A AGREGAR
Mitigations for CPU vulnerabilities	- Mitigate SPECTRE V1 hardware bug - Mitigate SPECTRE V2 hardware bug - Mitigate L1 Terminal Fault (L1TF) hardware bug	- Enable call trunks and call depth tracking debugging
Binary emulations	- X32 ABI for 64-bit mode	- IA32 emulation disabled by default
Virtualization	- Support for Microsoft Hyper-V emulation - AMD Secure Encrypted Virtualization (SEV) support	- Check that guests do not receive #VE exceptions - Prove KVM MMU correctness
Executable file formats	- Write ELF core dumps with partial segments	No podemos agregar una nueva, ya que todas estaban seleccionadas
Memory management options	- Enable VM event counters for /proc/vmstat - NUMA emulation	- Enable idle page tracking - Unaddressable device memory (GPU memory, …) - Anonymous VMA name support

3. Ahora, revisamos los módulos que vamos a quitar y agregar si estén o no habilitados. Para ello, buscando descubrimos que si nos vamos al help de las opciones en menú config, viene descrito el símbolo de la configuración:



4. Una vez que encontramos el símbolo de la característica, buscamos en la consola si está configurada en el kernel o no con el siguiente comando:

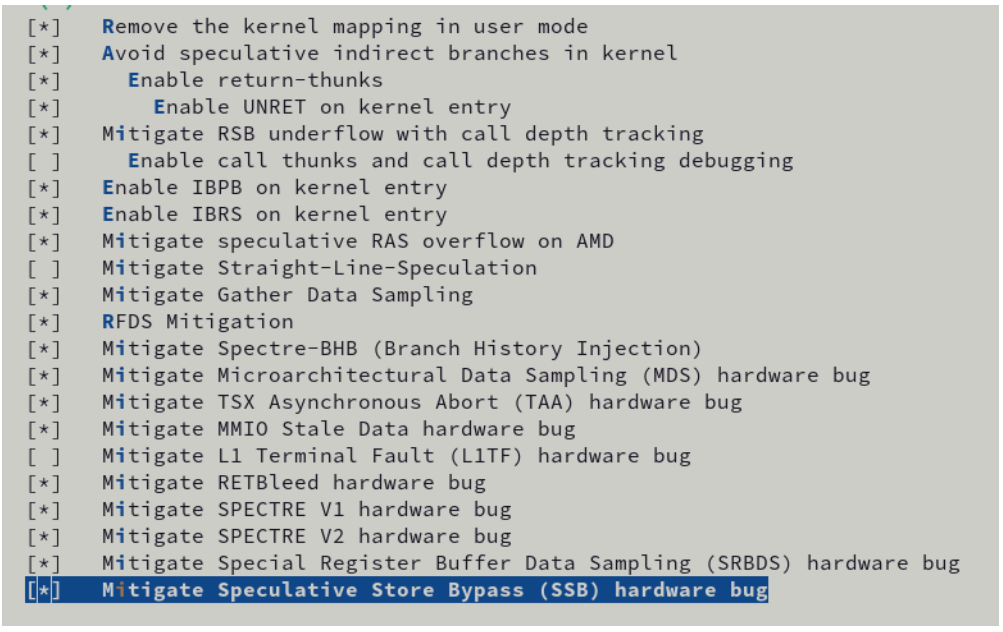
   `grep -i 'símbolo del módulo' /boot/config-$(uname -r)`

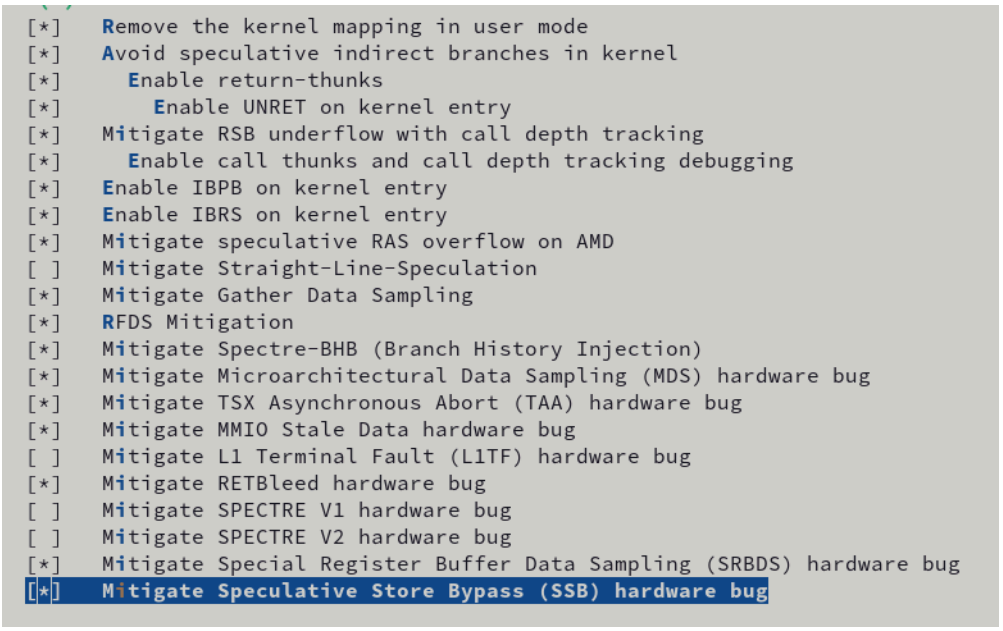
   Con esto, revisamos sección por sección si las características están habilitadas o deshabilitadas:

   1. Para **Mitigations for CPU** vulnerabilities buscamos con el comando:

      `grep -i 'MITIGATION_SPECTRE_V1' /boot/config-$(uname -r)`
      `grep -i 'MITIGATION_SPECTRE_V2' /boot/config-$(uname -r)`
      `grep -i 'MITIGATION_L1TF' /boot/config-$(uname -r)`
      `grep -i 'CALL_THUNKS_DEBUG' /boot/config-$(uname -r)`

```
yankees@yankees:/usr/src/linux-6.13$ grep -i 'MITIGATION_SPECTRE_V1' /boot/config-$(uname -r)
CONFIG_MITIGATION_SPECTRE_V1=y
yankees@yankees:/usr/src/linux-6.13$ grep -i 'MITIGATION_SPECTRE_V2' /boot/config-$(uname -r)
CONFIG_MITIGATION_SPECTRE_V2=y
yankees@yankees:/usr/src/linux-6.13$ grep -i 'MITIGATION_L1TF' /boot/config-$(uname -r)
CONFIG_MITIGATION_L1TF=y
yankees@yankees:/usr/src/linux-6.13$ grep -i 'CALL_THUNKS_DEBUG' /boot/config-$(uname -r)
# CONFIG_CALL_THUNKS_DEBUG is not set
yankees@yankees:/usr/src/linux-6.13$
```





2. Para **Binary emulations** buscamos con el comando:

```
grep -i 'X86_X32_ABI' /boot/config-$(uname -r)
grep -i 'IA32_EMULATION_DEFAULT_DISABLED' /boot/config-$(uname -r)
```

```
yankees@yankees:/usr/src/linux-6.13$ grep -i 'X86_X32_ABI' /boot/config-$(uname -r)
CONFIG_X86_X32_ABI=y
yankees@yankees:/usr/src/linux-6.13$ grep -i 'IA32_EMULATION_DEFAULT_DISABLED' /boot/config-$(uname -r)
# CONFIG_IA32_EMULATION_DEFAULT_DISABLED is not set
yankees@yankees:/usr/src/linux-6.13$
```

```
[*] IA32 Emulation
[*]   IA32 emulation disabled by default
[ ] x32 ABI for 64-bit mode
```

```
[*] IA32 Emulation
[ ]   IA32 emulation disabled by default
[*] x32 ABI for 64-bit mode
```

3. Para **Virtualization** buscamos con el comando:

```
grep -i 'KVM_HYPERV' /boot/config-$(uname -r)
grep -i 'KVM_AMD_SEV' /boot/config-$(uname -r)
grep -i 'KVM_INTEL_PROVE_VE' /boot/config-$(uname -r)
grep -i 'KVM_PROVE_MMU' /boot/config-$(uname -r)
```

```
yankees@yankees:/usr/src/linux-6.13$ grep -i 'KVM_HYPERV' /boot/config-$(uname -r)
CONFIG_KVM_HYPERV=y
yankees@yankees:/usr/src/linux-6.13$ grep -i 'KVM_AMD_SEV' /boot/config-$(uname -r)
CONFIG_KVM_AMD_SEV=y
yankees@yankees:/usr/src/linux-6.13$ grep -i 'KVM_INTEL_PROVE_VE' /boot/config-$(uname -r)
# CONFIG_KVM_INTEL_PROVE_VE is not set
yankees@yankees:/usr/src/linux-6.13$ grep -i 'KVM_PROVE_MMU' /boot/config-$(uname -r)
# CONFIG_KVM_PROVE_MMU is not set
yankees@yankees:/usr/src/linux-6.13$
```

```
--- Virtualization
<M>   Kernel-based Virtual Machine (KVM) support
[*]     Compile KVM with -Werror
[ ]     Enable support for KVM software-protected VMs
<M>     KVM for Intel (and compatible) processors support
[ ]       Check that guests do not receive #VE exceptions
[*]       Software Guard eXtensions (SGX) Virtualization
<M>     KVM for AMD processors support
[*]       AMD Secure Encrypted Virtualization (SEV) support
[*]     System Management Mode emulation
[*]     Support for Microsoft Hyper-V emulation
[ ]     Support for Xen hypercall interface
[ ]     Prove KVM MMU correctness
(4096) Maximum number of vCPUs per KVM guest
```

```
--- Virtualization
<M>   Kernel-based Virtual Machine (KVM) support
[*]     Compile KVM with -Werror
[ ]     Enable support for KVM software-protected VMs
<M>     KVM for Intel (and compatible) processors support
[*]       Check that guests do not receive #VE exceptions
[*]       Software Guard eXtensions (SGX) Virtualization
<M>     KVM for AMD processors support
[ ]       AMD Secure Encrypted Virtualization (SEV) support
[*]     System Management Mode emulation
[ ]     Support for Microsoft Hyper-V emulation
[ ]     Support for Xen hypercall interface
[*]     Prove KVM MMU correctness
(4096) Maximum number of vCPUs per KVM guest
```

4. Para **Executable file formats** buscamos con el comando:

```
grep -i 'CORE_DUMP_DEFAULT_ELF_HEADERS'
/boot/config-$(uname -r)
```

```
yankees@yankees:/usr/src/linux-6.13$ grep -i 'CORE_DUMP_DEFAULT_ELF_HEADERS' /boot/config-$(uname -r)
CONFIG_CORE_DUMP_DEFAULT_ELF_HEADERS=y
yankees@yankees:/usr/src/linux-6.13$
```

```
-*- Kernel support for ELF binaries
[*] Write ELF core dumps with partial segments
<*> Kernel support for scripts starting with #!
<M> Kernel support for MISC binaries
[*] Enable core dump support
```

```
-*- Kernel support for ELF binaries
[ ] Write ELF core dumps with partial segments
<*> Kernel support for scripts starting with #!
<M> Kernel support for MISC binaries
[*] Enable core dump support
```

5. Para **Memory management options** buscamos con el comando:

```
grep -i 'VM_EVENT_COUNTERS' /boot/config-$(uname -r)
grep -i 'NUMA_EMU' /boot/config-$(uname -r)
grep -i 'IDLE_PAGE_TRACKING' /boot/config-$(uname -r)
grep -i 'DEVICE_PRIVATE' /boot/config-$(uname -r)
grep -i 'ANON_VMA_NAME' /boot/config-$(uname -r)
```

```
yankees@yankees:/usr/src/linux-6.13$ grep -i 'VM_EVENT_COUNTERS' /boot/config-$(uname -r)
CONFIG_VM_EVENT_COUNTERS=y
yankees@yankees:/usr/src/linux-6.13$ grep -i 'NUMA_EMU' /boot/config-$(uname -r)
CONFIG_NUMA_EMU=y
yankees@yankees:/usr/src/linux-6.13$ grep -i 'IDLE_PAGE_TRACKING' /boot/config-$(uname -r)
# CONFIG_IDLE_PAGE_TRACKING is not set
yankees@yankees:/usr/src/linux-6.13$ grep -i 'DEVICE_PRIVATE' /boot/config-$(uname -r)
# CONFIG_DEVICE_PRIVATE is not set
yankees@yankees:/usr/src/linux-6.13$ grep -i 'ANON_VMA_NAME' /boot/config-$(uname -r)
# CONFIG_ANON_VMA_NAME is not set
yankees@yankees:/usr/src/linux-6.13$
```

```
[*] Support for paging of anonymous memory (swap)  --->
{M} N:1 compression allocator (zsmalloc)
[ ]   Export zsmalloc statistics
(8)   Maximum number of physical pages per-zspage
    Slab allocator options  --->
[*] Page allocator randomization
[ ] Disable heap randomization
-*- Sparse Memory virtual memmap
[*] Memory hotplug  --->
[*] Allow for balloon memory compaction/migration
-*- Allow for memory compaction
-*- Free page reporting
-*- Page migration
(5) Maximum scale factor of PCP (Per-CPU pageset) batch allocate/free
[*] Enable KSM for page merging
(65536) Low address space to protect from user allocation
[*] Enable recovery from hardware memory errors
<M>   HWPoison pages injector
[*] Transparent Hugepage Support  --->
[ ] Contiguous Memory Allocator
[*] Track memory changes
[*] Defer initialisation of struct pages to kthreads
v(+)
```

```
[*] Transparent Hugepage Support  --->
[ ] Contiguous Memory Allocator
[*] Track memory changes
[*] Defer initialisation of struct pages to kthreads
[ ] Enable idle page tracking
[*] Support DMA zone
-*- Support DMA32 zone
[*] Device memory (pmem, HMM, etc...) hotplug support
[ ] Unaddressable device memory (GPU memory, ...)
[*] Enable VM event counters for /proc/vmstat
[ ] Collect percpu memory statistics
[ ] Enable infrastructure for get_user_pages()-related unit tests
< > Enable a module to run time tests on dma_pool
-*- Enable memfd_create() system call
[*] Enable memfd_secret() system call
[ ] Anonymous VMA name support
[*] Enable userfaultfd() system call  --->
[*] Multi-Gen LRU
[ ]   Enable by default
[ ]   Full stats for debugging
[*] NUMA emulation
    Data Access Monitoring  --->
```

```
[*] Support for paging of anonymous memory (swap)  --->
{M} N:1 compression allocator (zsmalloc)
[ ]   Export zsmalloc statistics
(8)   Maximum number of physical pages per-zspage
      Slab allocator options  --->
[*] Page allocator randomization
[ ] Disable heap randomization
-*- Sparse Memory virtual memmap
[*] Memory hotplug  --->
[*] Allow for balloon memory compaction/migration
-*- Allow for memory compaction
-*- Free page reporting
-*- Page migration
(5) Maximum scale factor of PCP (Per-CPU pageset) batch allocate/free
[*] Enable KSM for page merging
(65536) Low address space to protect from user allocation
[*] Enable recovery from hardware memory errors
<M>   HWPoison pages injector
[*] Transparent Hugepage Support  --->
[ ] Contiguous Memory Allocator
[*] Track memory changes
[*] Defer initialisation of struct pages to kthreads
v(+)
```

```
[*] Transparent Hugepage Support  --->
[ ] Contiguous Memory Allocator
[*] Track memory changes
[*] Defer initialisation of struct pages to kthreads
[*] Enable idle page tracking
[*] Support DMA zone
-*- Support DMA32 zone
[*] Device memory (pmem, HMM, etc...) hotplug support
[*] Unaddressable device memory (GPU memory, ...)
[ ] Enable VM event counters for /proc/vmstat
[ ] Collect percpu memory statistics
[ ] Enable infrastructure for get_user_pages()-related unit tests
< > Enable a module to run time tests on dma_pool
-*- Enable memfd_create() system call
[*] Enable memfd_secret() system call
[*] Anonymous VMA name support
[*] Enable userfaultfd() system call  --->
[*] Multi-Gen LRU
[ ]   Enable by default
[ ]   Full stats for debugging
[ ] NUMA emulation
    Data Access Monitoring  --->
```

Identificación del kernel

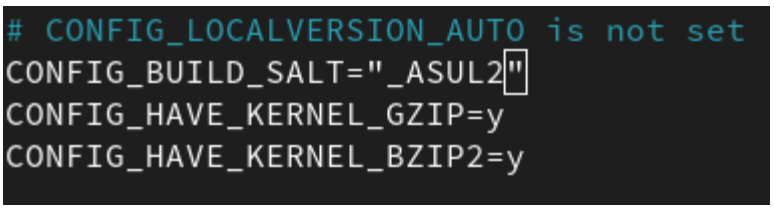
1. Usamos:

   sudo nano .config

   Para poder agregar:

   CONFIG_BUILD_SALT="_ASUL2”

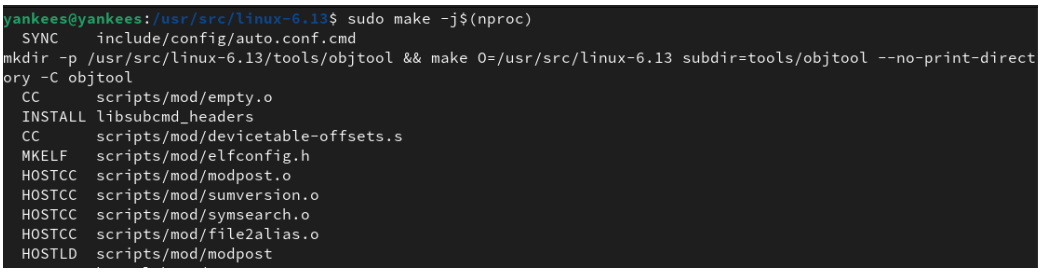
   Y así poder identificar nuestro kernel.

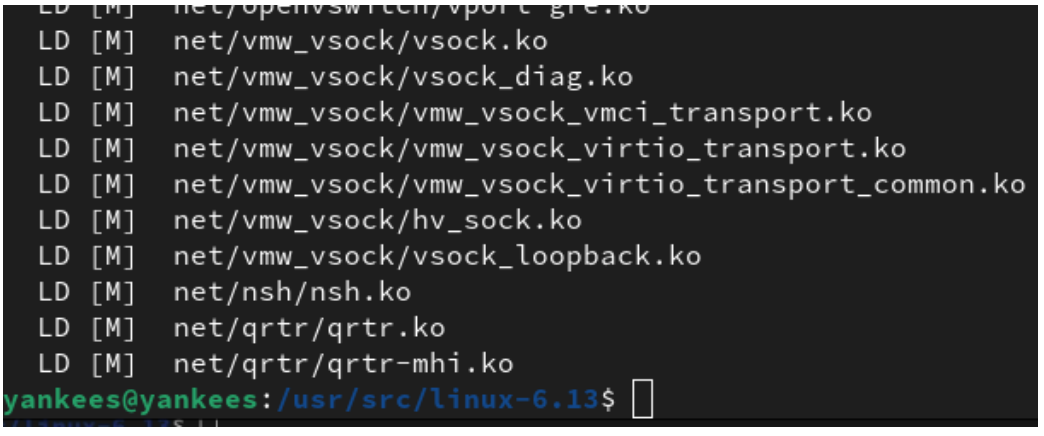


Compilación del kernel

1. Compilamos el kernel y los módulos con el comando:

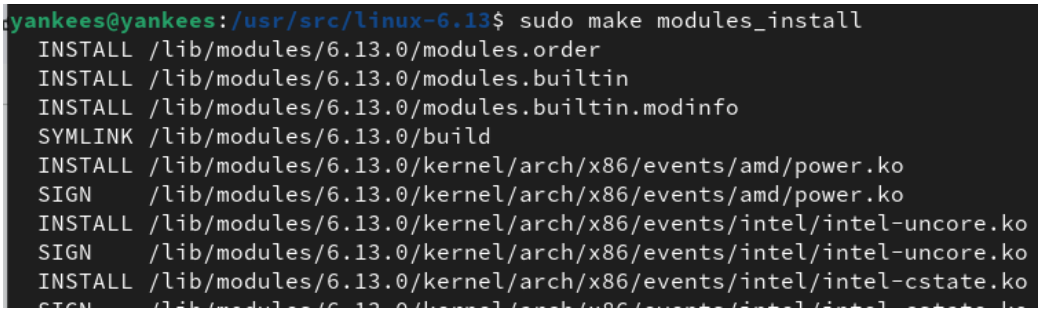
   sudo make -j$(nproc)





2. Después de unas horas y terminar de compilar, ahora tenemos que compilar los módulos con el comando:

   sudo make modules_install



```
  SIGN    /lib/modules/6.13.0/kernel/net/vmw_vsock/vsock_loc
  INSTALL /lib/modules/6.13.0/kernel/net/nsh/nsh.ko
  SIGN    /lib/modules/6.13.0/kernel/net/nsh/nsh.ko
  INSTALL /lib/modules/6.13.0/kernel/net/qrtr/qrtr.ko
  SIGN    /lib/modules/6.13.0/kernel/net/qrtr/qrtr.ko
  INSTALL /lib/modules/6.13.0/kernel/net/qrtr/qrtr-mhi.ko
  SIGN    /lib/modules/6.13.0/kernel/net/qrtr/qrtr-mhi.ko
  DEPMOD  /lib/modules/6.13.0
yankees@yankees:/usr/src/linux-6.13$
```

Instalación del kernel

1. Instalamos el kernel con el comando:

`sudo make install`

```
yankees@yankees:/usr/src/linux-6.13$ sudo make install
  INSTALL /boot
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 6.13.0 /boot/vmlinuz-6.13.0
update-initramfs: Generating /boot/initrd.img-6.13.0
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 6.13.0 /boot/vmlinuz-6.13.0
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.13.0
Found initrd image: /boot/initrd.img-6.13.0
Found linux image: /boot/vmlinuz-6.13.0.old
Found initrd image: /boot/initrd.img-6.13.0
Found linux image: /boot/vmlinuz-6.1.0-29-amd64
Found initrd image: /boot/initrd.img-6.1.0-29-amd64
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
Adding boot menu entry for UEFI Firmware Settings ...
done
```

Actualización de la configuración

1. Actualizamos la configuración del grub con el comando:

`sudo update-grub`

```
yankees@yankees:/usr/src/linux-6.13$ sudo update-grub
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.13.0
Found initrd image: /boot/initrd.img-6.13.0
Found linux image: /boot/vmlinuz-6.13.0.old
Found initrd image: /boot/initrd.img-6.13.0
Found linux image: /boot/vmlinuz-6.1.0-29-amd64
Found initrd image: /boot/initrd.img-6.1.0-29-amd64
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
Adding boot menu entry for UEFI Firmware Settings ...
done
```

2. Después, reiniciamos el sistema con el comando:

`sudo reboot`

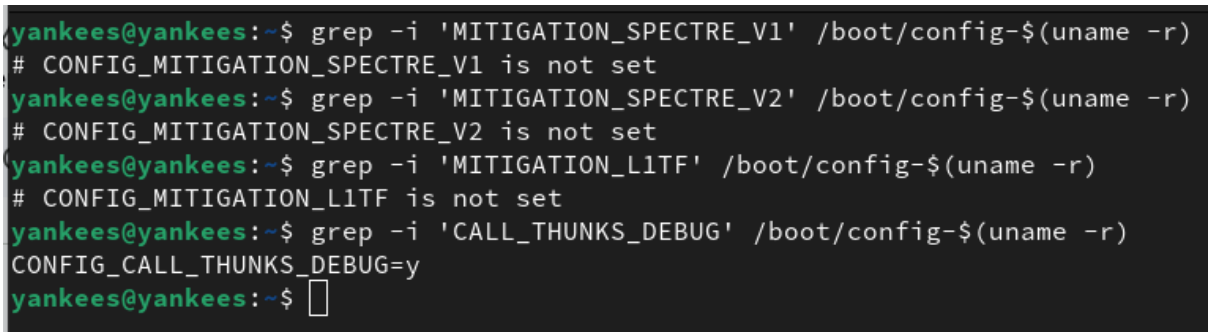
```
yankees@yankees:/usr/src/linux-6.13$ sudo reboot

Broadcast message from root@yankees on pts/2 (Sat 2025-03-29 15:28:32 CST):

The system will reboot now!

yankees@yankees:/usr/src/linux-6.13$ Connection to 192.168.23.46 closed by remote host.
```

Verificación del sistema

1. Por último, revisamos que los módulos que agregamos y quitamos si se hayan aplicado. Para ello, volvemos a usar los comandos del punto 4 de la sección de asignación y eliminación de módulos:

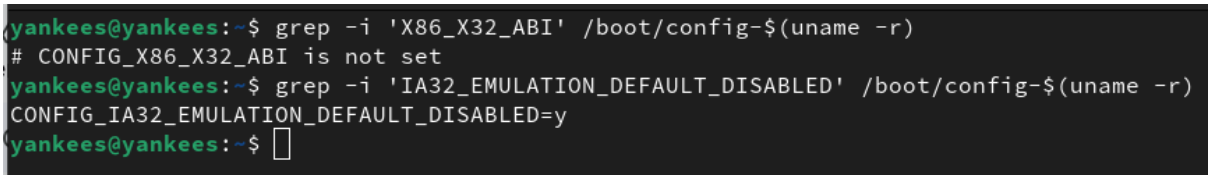
    1. Para **Mitigations for CPU** vulnerabilities buscamos con el comando:

        ```
        grep -i 'MITIGATION_SPECTRE_V1' /boot/config-$(uname -r)
        grep -i 'MITIGATION_SPECTRE_V2' /boot/config-$(uname -r)
        grep -i 'MITIGATION_L1TF' /boot/config-$(uname -r)
        grep -i 'CALL_THUNKS_DEBUG' /boot/config-$(uname -r)
        ```



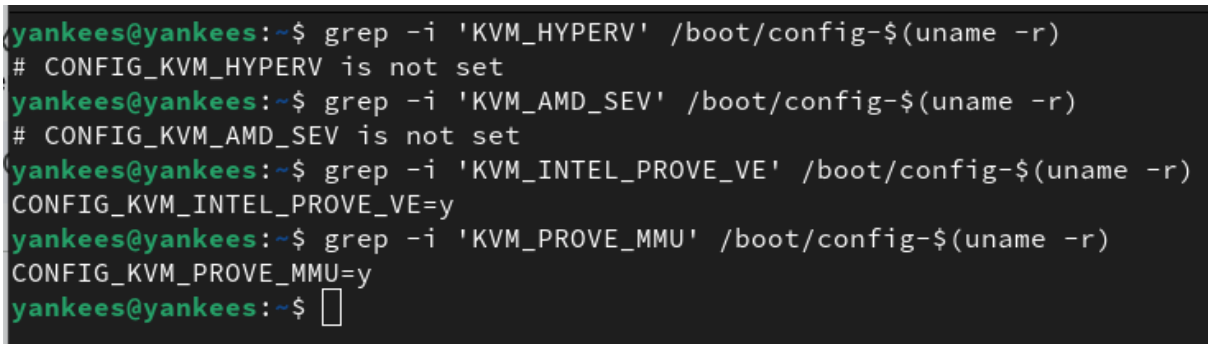
    2. Para **Binary emulations** buscamos con el comando:

        ```
        grep -i 'X86_X32_ABI' /boot/config-$(uname -r)
        grep -i 'IA32_EMULATION_DEFAULT_DISABLED'
                /boot/config-$(uname -r)
        ```



    3. Para **Virtualization** buscamos con el comando:

        ```
        grep -i 'KVM_HYPERV' /boot/config-$(uname -r)
        grep -i 'KVM_AMD_SEV' /boot/config-$(uname -r)
        grep -i 'KVM_INTEL_PROVE_VE' /boot/config-$(uname -r)
        grep -i 'KVM_PROVE_MMU' /boot/config-$(uname -r)
        ```



4. Para **Executable file formats** buscamos con el comando:
   `grep -i 'CORE_DUMP_DEFAULT_ELF_HEADERS' /boot/config-$(uname -r)`

```
yankees@yankees:~$ grep -i 'CORE_DUMP_DEFAULT_ELF_HEADERS' /boot/config-$(uname -r)
# CONFIG_CORE_DUMP_DEFAULT_ELF_HEADERS is not set
yankees@yankees:~$
```

5. Para **Memory management options** buscamos con el comando:
   `grep -i 'VM_EVENT_COUNTERS' /boot/config-$(uname -r)`
   `grep -i 'NUMA_EMU' /boot/config-$(uname -r)`
   `grep -i 'IDLE_PAGE_TRACKING' /boot/config-$(uname -r)`
   `grep -i 'DEVICE_PRIVATE' /boot/config-$(uname -r)`
   `grep -i 'ANON_VMA_NAME' /boot/config-$(uname -r)`

```
yankees@yankees:~$ grep -i 'VM_EVENT_COUNTERS' /boot/config-$(uname -r)
# CONFIG_VM_EVENT_COUNTERS is not set
yankees@yankees:~$ grep -i 'NUMA_EMU' /boot/config-$(uname -r)
# CONFIG_NUMA_EMU is not set
yankees@yankees:~$ grep -i 'IDLE_PAGE_TRACKING' /boot/config-$(uname -r)
CONFIG_IDLE_PAGE_TRACKING=y
yankees@yankees:~$ grep -i 'DEVICE_PRIVATE' /boot/config-$(uname -r)
CONFIG_DEVICE_PRIVATE=y
yankees@yankees:~$ grep -i 'ANON_VMA_NAME' /boot/config-$(uname -r)
CONFIG_ANON_VMA_NAME=y
yankees@yankees:~$
```

Características del kernel

- Describe todas las características que agregaste o eliminaste del kernel y explica por qué lo hiciste, detallando cómo pueden impactar en el rendimiento, compatibilidad o seguridad de tu sistema.

  Las características que eliminamos fueron las siguientes:

  - Para **Mitigations for CPU:**
    - *Mitigate SPECTRE V1 hardware bug*
      - Impacto: SPECTRE V1 explota la ejecución especulativa de los procesadores, lo que permite a un atacante leer datos de la memoria que no deberían ser accesibles. Eliminar esta mitigación puede aumentar el riesgo de vulnerabilidad ante ataques de este tipo, pero también puede mejorar el rendimiento al eliminar las comprobaciones adicionales de seguridad. Y ya que pensamos que no vamos a recibir tantos

ataques (al menos a la computadora del laboratorio) pensamos que es opción decente quitar este módulo.

- *Mitigate SPECTRE V2 hardware bug*
  - Impacto: Similar al V1, el SPECTRE V2 afecta la ejecución especulativa en los procesadores. Al eliminar esta mitigación, se podría mejorar el rendimiento, pero también se aumenta el riesgo de posibles explotaciones de seguridad relacionadas con la especulación en el procesador. De manera similar al módulo anterior, como no esperamos ser atacados pensamos que este módulo no es tan necesario y al quitarlo aumentamos el rendimiento del equipo.
  -
- *Mitigate L1 Terminal Fault (L1TF) hardware bug*
  - Impacto: El L1TF es un error de hardware que afecta a los procesadores de Intel y puede permitir que los atacantes accedan a la memoria sensible. Su eliminación podría mejorar el rendimiento, pero comprometería la seguridad al no mitigar esta vulnerabilidad. Quitamos este módulo ya que, igual que los dos anteriores, está relacionado a ataques por lo tanto no pensamos que sea tan importante, si el equipo se usará más en redes públicas este modelo podría ser muy útil.

- Para **Binary emulations:**
  - *X32 ABI for 64-bit mode*
    - Impacto: El ABI X32 permite que aplicaciones de 32 bits se ejecuten en un sistema de 64 bits. Eliminar esta característica puede reducir la compatibilidad con aplicaciones de 32 bits, pero también puede simplificar la arquitectura y mejorar el rendimiento al no emular entornos de 32 bits. La principal razón para quitar este módulo es que la mayoría de las aplicaciones hoy en día tienen versiones en 64 bits (a excepción de aplicaciones antiguas) entonces pensamos que un mejor rendimiento puede ser más útil.

- Para **Virtualization:**
  - *Support for Microsoft Hyper-V emulation*
    - Impacto: El soporte para la emulación de Hyper-V permite ejecutar máquinas virtuales de Windows en plataformas que no son de Microsoft. Eliminar esta opción podría mejorar el rendimiento y reducir la complejidad, pero reduciría la compatibilidad con las soluciones de virtualización basadas en

Hyper-V. Quitamos este módulo ya que no creemos que bajamos a usar máquinas virtuales en el equipo del laboratorio y menos una máquina virtual de Windows.

- *AMD Secure Encrypted Virtualization (SEV) support*
  - Impacto: SEV proporciona cifrado de memoria para máquinas virtuales en plataformas AMD. Eliminarlo puede hacer que el sistema sea menos seguro, pero puede mejorar el rendimiento de las máquinas virtuales al eliminar las comprobaciones de seguridad adicionales. Otro módulo que tiene que ver con máquinas virtuales pero este es con su memoria, como no creemos usar máquinas virtuales entonces no lo vemos tan útil, además el cifrado de la memoria de las máquinas virtuales tampoco sería indispensable si las usamos.

- Para **Executable file formats**:
  - *Write ELF core dumps with partial segments*
    - Impacto: La eliminación de esta característica impide escribir volcados de memoria con segmentos parciales, lo que podría dificultar la depuración, pero también podría mejorar la seguridad al reducir la cantidad de información sensible almacenada en los volcados. Este módulo lo quitamos ya que como todos los módulos de esta sección estaban actividades entonces decidimos quitar el que menos pensamos que afectará al funcionamiento del equipo.

- Para **Memory management options**:
  - *Enable VM event counters for /proc/vmstat*
    - Impacto: Esta opción habilita el seguimiento de eventos de memoria en tiempo real. Eliminarla podría reducir la sobrecarga del sistema, pero limitaría las métricas detalladas sobre el rendimiento de la memoria. Ya que en la computadora del laboratorio no usamos programas para monitorear memoria en tiempo real, si fuéramos a monitorear la memoria seria muy util activarlo pero eso lo podemos hacer dependiendo de lo que necesitemos.
  - *NUMA emulation*
    - Impacto: NUMA (Non-Uniform Memory Access) emula configuraciones de memoria no uniforme. Al eliminarlo, se podría mejorar el rendimiento en sistemas que no requieren esta funcionalidad, pero podría limitar la escalabilidad y el

rendimiento en máquinas con múltiples procesadores. Quitamos este módulo ya que creemos que el hardware del equipo no es NUMA y no creemos usar cosas relacionadas a NUMA (es la primera vez que oímos ese término y no sabemos muchas cosas donde se use).

Las características que agregamos fueron las siguientes:

- Para **Mitigations for CPU:**
    - *Enable call trunks and call depth tracking debugging*
        - Impacto: Habilitar el rastreo de profundidad de llamadas y troncos puede ayudar en la depuración y análisis de las rutas de ejecución del código. Esto puede tener un impacto en el rendimiento debido al costo adicional de seguimiento, pero mejora la trazabilidad de los problemas. Activamos este módulo ya que pensamos que puede ser útil en situaciones donde tengamos errores de stack overflow o aspectos recursivos y así poder intentar resolverlos.
- Para **Binary emulations:**
    - *IA32 emulation disabled by default*
        - Impacto: Al deshabilitar la emulación de IA32 de manera predeterminada, se mejora el rendimiento al evitar la necesidad de ejecutar código de 32 bits en sistemas de 64 bits, aunque reduce la compatibilidad con aplicaciones antiguas. Este módulo está algo relacionado al otro de cosas de 32 bits, como pensamos no usar programas de 32 bits entonces este módulo es útil activarlo.
- Para **Virtualization:**
    - *Check that guests do not receive #VE exceptions*
        - Impacto: Esta característica ayuda a garantizar que las máquinas virtuales no reciban excepciones inesperadas. Agregarla mejora la estabilidad y fiabilidad de las máquinas virtuales, aunque podría introducir una ligera sobrecarga al verificar estas condiciones. En este módulo no tuvimos tantas razones para activarlo, ya que pensamos que no usaremos máquinas virtuales, pero como evita que los huéspedes reciban excepciones entonces pensamos que suena útil.
    - *Prove KVM MMU correctness*

        - Impacto: Esta característica mejora la precisión de la gestión de memoria en máquinas virtuales utilizando KVM (Kernel-based Virtual Machine). Puede mejorar la estabilidad de las VM, pero también puede tener un pequeño impacto en el rendimiento debido a las verificaciones adicionales. La principal razón por la que activamos este módulo es que tiene que ver con memoria de máquinas virtuales y es uno de los temas que hemos visto en clase, eso es lo único por lo que pensamos agregar este módulo.

- Para **Executable file formats**:

  No pudimos agregar una nueva, ya que todas estaban seleccionadas.

- Para **Memory management options**:

    - *Enable idle page tracking*

        - Impacto: Habilitar el seguimiento de páginas inactivas puede mejorar la eficiencia del sistema al gestionar mejor las páginas de memoria que no están siendo utilizadas. Sin embargo, podría aumentar la sobrecarga de administración de memoria. Este módulo nos pareció bueno agregarlo ya que puede mejorar nuestro monitoreo de la memoria, ya sea para identificar problemas o poder liberar memoria.

    - *Unaddressable device memory (GPU memory, …)*

        - Impacto: Agregar soporte para memoria no direccionable de dispositivos puede permitir un mejor manejo de la memoria de hardware, como la memoria de la GPU, pero podría aumentar la complejidad y el costo de administración de memoria. Pensamos que este módulo puede ser algo útil, pero no estamos tan seguros, una de las razones por la que escogimos este módulo es que está relacionada a memoria no direccionable y pensamos que esto puede ser importante dependiendo de los dispositivos.

    - *Anonymous VMA name support*

        - Impacto: Añadir soporte para nombres de VMA (Áreas de memoria virtual) anónimos puede mejorar la capacidad de diagnóstico y trazabilidad, pero puede añadir cierta sobrecarga al sistema. Activamos este módulo ya que tiene que ver con la memoria virtual, la cual creemos que puede ser algo importante ya que hay algunas veces donde la usamos, por ejemplo al usar malloc().

- Investiga también aquellas características que te parecieron más interesantes (dentro de las secciones mencionadas), no necesariamente aquellas que modificaste y explica su función.

    - Para **Mitigations for CPU:**

        - *Mitigate Meltdown hardware bug*

          Función: Mitiga una vulnerabilidad en los procesadores modernos que permite a los atacantes leer información sensible de la memoria, como contraseñas o claves criptográficas, a través de la ejecución especulativa.

          Ventajas: Aumenta la seguridad de la máquina frente a ataques de Meltdown.

          Desventajas: Impacta en el rendimiento, ya que requiere controles adicionales para evitar la vulnerabilidad.

        - *Remove the kernel mapping in user mode*

          Función: Separar las tablas de páginas utilizadas por el modo usuario y el modo kernel, así evitar que procesos en modo usuario puedan acceder a la memoria del kernel mediante vulnerabilidades como la del módulo anterior.

          Ventajas: Aumenta la seguridad ante amenazas que usen la memoria del kernel y ayuda a tener mayor protección en entornos donde hay varios usuarios que comparten recursos.

          Desventajas: El uso de memoria es mayor ya que se necesita un conjunto de tablas de páginas para el modo usuario y otro para el modo kernel, además el rendimiento del equipo puede ser algo menor debido a esto, más si se realizan varias llamadas del sistema.

    - Para **Binary emulations:**

        - *IA32 emulation (IA-32 ABI)*

          Función: Emula las aplicaciones de 32 bits en sistemas de 64 bits, lo que permite la compatibilidad con programas antiguos o que no se han actualizado a 64 bits.

          Ventajas: Aumenta la compatibilidad con software de 32 bits, lo que es útil para aplicaciones legadas

Desventajas: El uso de emulaciones aumenta la complejidad y puede generar una penalización de rendimiento debido a la emulación del conjunto de instrucciones.

- Para **Virtualization:**
    - *Support for Microsoft Hyper-V emulation*

      Función: Permite que el sistema operativo emule el entorno Hyper-V, lo que facilita la ejecución de máquinas virtuales Windows en sistemas que no son de Microsoft.

      Ventajas: Mejora la compatibilidad con máquinas virtuales en plataformas que no usan Hyper-V nativamente.

      Desventajas: Introduce una sobrecarga en términos de recursos y complejidad, lo que puede impactar el rendimiento en entornos virtualizados.

    - *Compile KVM with -Werror*

      Función: Cuando el código del componente KVM (Kernel-based Virtual Machine) se compila y obtiene alguna advertencia (warning) del compilador se van a tomar como errores, de esta manera si hay una advertencia durante la compilación del componente entonces el proceso fallará.

      Ventajas: Evita posibles bugs o errores que pueden surgir por las advertencias y garantiza que el componente sirva muy similar en diferentes entornos, ya que se busca compilar todo bien.

      Desventajas: El proceso de compilación puede ser algo más tardado ya que no tienen que existir advertencias en el componente y también dificulta la compilación del kernel si alguien quiere usar módulos creados por ellos mismos, ya que pueden generar advertencias.

- Para **Executable file formats**:
    - *Support for large ELF binaries support*

      Función: Permite la ejecución y manipulación de archivos ELF de gran tamaño, como aquellos que superan los 2 GB en sistemas de 32 bits.

      Ventajas: Permite trabajar con archivos ejecutables y volcados de memoria de gran tamaño sin tener que realizar modificaciones.

Desventajas: Puede requerir más recursos del sistema y es menos relevante en sistemas con archivos más pequeños o donde no se trabajan con grandes binarios.

- *Kernel support for MISC binaries*

  Función: Permite que el kernel puede ejecutar binarios con formatos diferentes al nativo, por ejemplo permite ejecutar scripts sin !# al inicio o binarios de otros sistemas operativos.

  Ventajas: Permite ejecutar binarios de otros sistemas operativos sin tener que hacer cambios en el kernel y esto permite una mayor ayuda en el desarrollo.

  Desventajas: Puede ser algo inseguro ya que usuarios no confiables podrían ejecutar binarios peligrosos y no es necesario este módulo si no se piensa ejecutar binarios no nativos.

- Para **Memory management options**:

  - *Transparent Huge Pages*

    Función: Proporciona soporte para páginas de memoria de gran tamaño de forma transparente, sin necesidad de que el usuario las configure manualmente.

    Ventajas: Mejora el rendimiento de aplicaciones que usan grandes cantidades de memoria, ya que reduce la sobrecarga de la gestión de páginas.

    Desventajas: Puede ser incompatible con ciertas aplicaciones que requieren un control más preciso sobre la memoria y, en algunos casos, puede causar problemas de rendimiento si no se usa correctamente.

  - *Free page reporting*

    Función: Permite al kernel reportar al hipervisor cuáles son las páginas de memoria que están libres, así poder usarlas en máquinas virtuales o con el host.

    Ventajas: Mejora uso de la memoria en virtualización y en el sistema en general, y permite poder la optimización del uso de la memoria de manera automática.

    Desventajas: Tiene un costo el escanear y reportar las páginas libres (pero es pequeño) y requiere que el hipervisor y herramientas de virtualización soporte esta función.

Preguntas

1. **Investiga para qué sirve el comando make defconfig, ¿Cuándo se utiliza?**

   El comando make defconfig sirve para generar una nueva configuración (un .config) con las opciones predeterminadas proporcionadas por el archivo defconfig que nos da ARCH o la arquitectura con la que estamos trabajando. Este comando se puede utilizar cuando queremos una configuración estándar para cierta arquitectura y a partir de esta configuración la podemos ir modificando dependiendo de lo que necesitemos, otra ocasión cuando se usa es cuando necesitamos recuperar la configuración que venía con el código fuente del kernel.

2. **¿Cómo se carga un módulo dinámicamente? Da un ejemplo.**

   Una manera de realizar esto es con el comando “insmod” el cual carga un módulo, un ejemplo de uso sería “sudo insmod <módulo>”.
   Otra forma para cargar un módulo es con el comando “modprobe”, este comando sirve para cargar y quitar módulos, un ejemplo de uso sería “sudo modprobe <módulo>”, algo importante es que si el módulo tiene dependencias estas también serán cargadas.
   Estas son algunas maneras para poder cargar módulos, tal vez la más recomendada para usar es “modprobe”, ya que “insmod” es más simple y trivial.

3. **¿Cómo se remueve un módulo que está cargado en el kernel? Da un ejemplo.**

   Una manera de realizar esto es con el comando “rmmod” el cual remueve un módulo, un ejemplo de uso sería “sudo rmmod <módulo>”.
   Otra forma para remover un módulo es con el comando “modprobe”, el mismo comando de la pregunta anterior pero ahora usando la bandera “-r”, un ejemplo de uso sería “sudo modprobe -r <módulo>”, algo importante es que si el módulo tiene dependencias que ya no seran usadas entonces estas también serán removidas.
   Estas son algunas maneras para poder cargar módulos, tal vez la más recomendada para usar es “modprobe”, ya que “rmmod” es más simple y trivial (algo similar a la comparación entre “insmod” y “modprobe”).

4. **¿Cómo puedes hacer que un módulo se cargue automáticamente al arrancar el sistema?**

   Una manera para realizar esto es creando un archivo con terminación “.conf” en “/etc/modules-load.d/” y dentro del nuevo archivo agregamos el nombre del módulo. Por ejemplo, creamos un archivo “mimodulo.conf” dentro de “/etc/modules-load.d/” (suponiendo que el módulo se llama mimodulo) y luego dentro del archivo ponemos “mimodulo”.

Esto sirve ya que systemd-modules-load.service lee archivos de ese directorio y obtiene los módulos del kernel que se tienen que cargar durante el arranque.

Referencias

Las referencias que se usaron fueron las siguientes:

- (2025). Kernel.org. https://www.kernel.org/doc/makehelp.txt
- insmod(8): insert module into Kernel - Linux man page. (2025). Die.net. https://linux.die.net/man/8/insmod
- modprobe(8): add/remove modules from Kernel - Linux man page. (2025). Die.net. https://linux.die.net/man/8/modprobe
- rmmod(8): remove module from Kernel - Linux man page. (2025). Die.net. https://linux.die.net/man/8/rmmod
- can. (2019, November 27). can linux rmmod command remove all modules? Super User. https://superuser.com/questions/1505496/can-linux-rmmod-command-remove-all-modules
- modules-load.d. (2025). Freedesktop.org. https://www.freedesktop.org/software/systemd/man/latest/modules-load.d.html
- systemd-modules-load.service. (2025). Freedesktop.org. https://www.freedesktop.org/software/systemd/man/latest/systemd-modules-load.service.html

Conclusión, comentarios y observaciones

La realización de esta práctica fue algo laboriosa, ya que investigamos mucho sobre cómo afectaría quitar o agregar cierta característica para no provocar fallos en el sistema después de compilar el kernel con esos cambios. Después, lo único difícil fue encontrar el comando para ver si los módulos estaban activos o no. Para ello, usamos el símbolo que se puede ver escribiendo H en la característica para hacer un grep sobre ese valor.

A pesar de todo esto, fue muy interesante ver que hay muchas cosas que podemos personalizar y que son muy útiles dependiendo de las actividades que se realicen con Linux. Si bien, es conocido que Linux es muy personalizable, nunca habíamos indagado en las características del kernel, lo cual nos ayudó a comprender mejor cómo funciona este sistema operativo.