

Sistema de comunicaciones seguras con segmentación virtual de dominios

Avance de proyecto

Alberto Daniel Lange

Dirección: Juan Ignacio Vaccarezza
Codirección: Santiago Pérez Ghiglia

Ingeniería en Telecomunicaciones
Instituto Balseiro

26 de febrero de 2025



① Introducción

② Revisión bibliográfica

③ Desarrollo

④ Conclusiones

① Introducción

② Revisión bibliográfica

③ Desarrollo

④ Conclusiones

Contexto

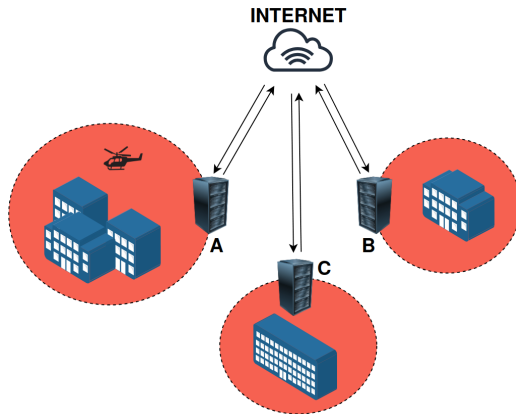


Figura 1: Esquema simplificado de operación del sistema.

Progreso alcanzado anteriormente

- Definición y validación de arquitectura lógica en laboratorio virtual.

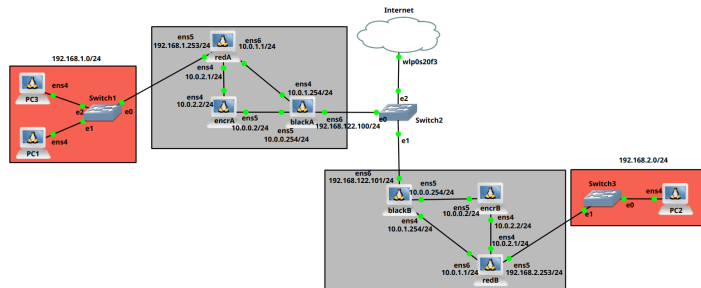


Figura 2: Implementación de la propuesta de solución en GNS3.

Pendientes

- Validación del encriptador en ambiente virtualizado.
- Implementación sobre hardware.

① Introducción

② Revisión bibliográfica

③ Desarrollo

④ Conclusiones

seL4

- Microkernel de código abierto.
- Formalmente probado.
- Hipervisor tipo 1.
- Aislamiento garantizado entre componentes.

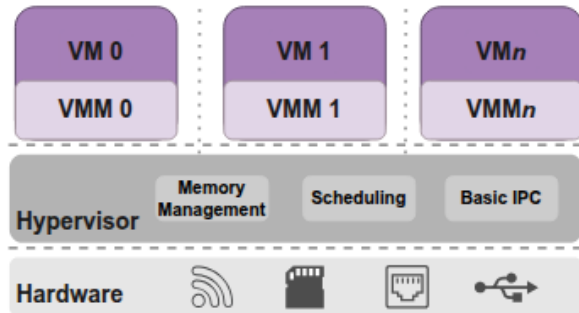


Figura 3: Arquitectura de seL4. (algo así, otra imagen)

- **Component Architecture for microkernel-based Embedded Systems.**
- Framework de desarrollo para seL4.
- Arquitectura basada en componentes.
- Comunicación mediante interfaces bien definidas.

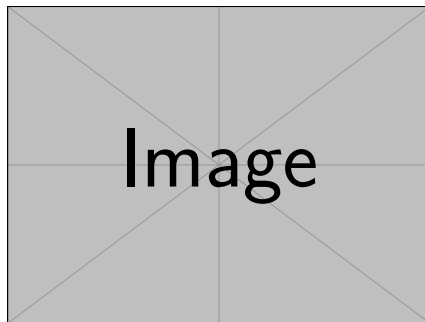


Figura 4: Arquitectura de componentes CAmkES.

Modelo minimal_64

- Ejemplo básico de CAmkES.
- Implementación mínima de una VM para arquitectura de 64 bits.
- Punto de partida para el desarrollo.

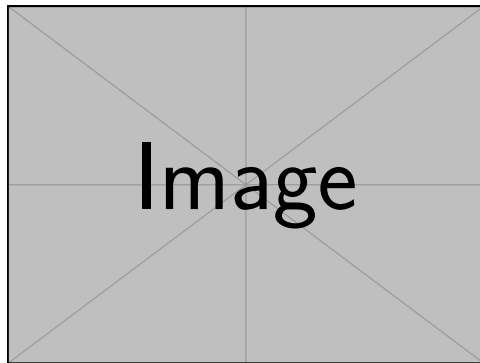


Figura 5: Arquitectura minimal_64.

Modelo zmq_samples

- Comunicación entre máquinas virtuales usando ZeroMQ.
- Base para la implementación del encriptador.

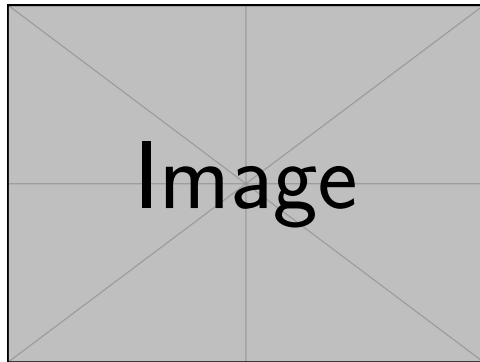


Figura 6: Arquitectura zmq_samples.

① Introducción

② Revisión bibliográfica

③ Desarrollo

④ Conclusiones

Estrategia de modelos en entornos virtualizados

- **¿Qué?:** Realizar modelos que validen progresivamente los componentes desarrollados.
- **¿Para qué?:**
 - Ligar problemas concretos a cada modelo y resolverlos de forma independiente.
 - Implementar un encriptador funcional en un entorno virtualizado como paso previo a su despliegue en hardware.

Modelo I: Introduciendo la arquitectura lógica

- Validar la arquitectura lógica de tres VMs.
- Funcionalidad *split-tunneling*.

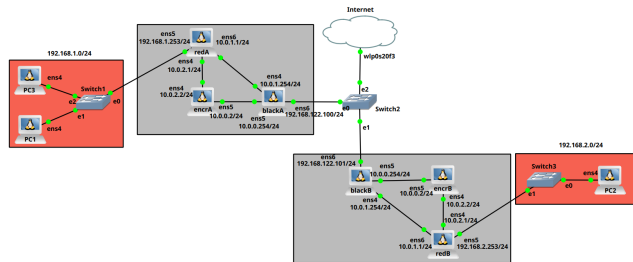


Figura 7: Arquitectura lógica en GNS3.

Modelo II: comunicando sitios con WireGuard

- Validación del kernel Linux 4.9 con soporte para WireGuard.
- Obtener parámetros PCI necesarios para el passthrough de la interfaz de red.

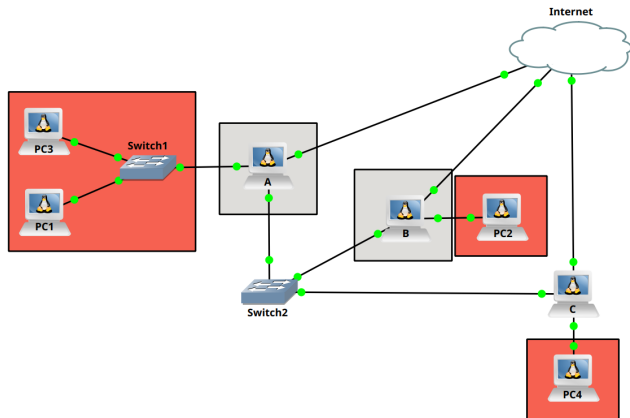


Figura 8: Arquitectura lógica en GNS3.

Modelo II: comunicando sitios con WireGuard - Sistema operativo VMs

- Buildroot como sistema de archivos.
- Kernel Linux 4.9.337.
- Parche de compatibilidad WireGuard con Linux ≤ 5.6 .

Modelo III: utilizando seL4 como hipervisor

- Validación del passthrough de hardware y la compatibilidad del kernel Linux 4.9 con seL4.

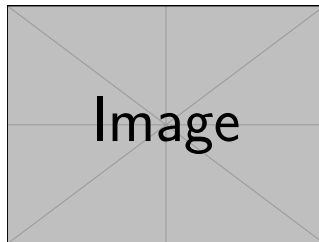


Figura 9: Algo

Modelo III: utilizando seL4 como hipervisor - Passthrough de hardware

CAMkES actúa de interfaz para la configuración del passthrough de dispositivos PCI a una VM a través de los Base Address Registers (BARs) e interrupciones.

Ejemplo de configuración en CAMkES:

```
vm0.vm_ioports = [start, end];  
vm0.pci_devices = [bus, device, function, memory];  
vm0.vm_irqs = [source, dest];
```

Modelo III: utilizando seL4 como hipervisor - Gestión de memoria VMs

`simple_untypedN_pool`: prealocación de memoria para las VMs.

`heap_size`: tamaño del heap del VMM.

`guest_ram_mb`: tamaño de la RAM asignada a cada VM.

Modelo IV: implementando el encriptador en seL4

- Solución completa como paso previo a la implementación en hardware.

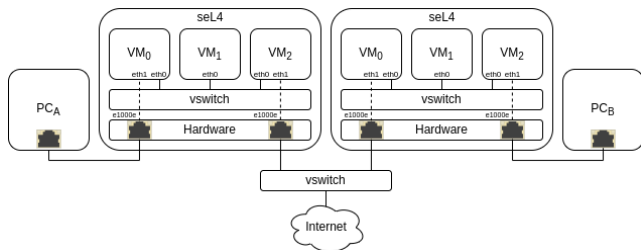


Figura 10: Algo.

Modelo IV: Encriptador en seL4 - Comunicación entre VMs

¿Cómo funciona?

Virtio-Net en VM

- El host implementa un dispositivo PCI virtual (virtio-net).
- La VM encuentra este dispositivo y carga el driver virtio-net.
- En la transmisión, la VM escribe los datos a enviar en un buffer asociado a una cola del hipervisor. El VMM notifica al host que hay datos listos en la cola (virtqueue).

Virtio-Net en host

- El host lee el paquete de la virtqueue y lo copia a un buffer dentro del espacio de memoria de la VM destino.
- El host inyecta una interrupción en la VM destino para notificarle que hay datos disponibles.

Modelo IV: Encryptador en seL4 - Integración

The image displays a complex network security configuration and monitoring environment. It includes configuration files for network interfaces, a packet capture showing network traffic, and a terminal window showing the execution of network commands.

Configuration Files (Left Panel):

```
valid_lft forever preferred_lft forever
3: ipddp0: <BROADCAST,MULTICAST,NOARP> mtu 585 qdisc noop state DOWN group default qlen 1000
   link/lp/ddp 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
4: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default q0
   link/ether 00:00:00:00:00:02 brd ff:ff:ff:ff:ff:ff
   inet 100.64.0.2/24 scope global eth1
   valid_lft forever preferred_lft forever
   inet6 fe80::200:ff:fe00:2/64 scope link
   valid_lft forever preferred_lft forever
5: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1
   link/lp/lp 0.0.0.0 brd 0.0.0.0
6: gre0@NONE: <NOARP> mtu 1476 qdisc noop state DOWN group default qlen 1
   link/gre 0.0.0.0 brd 0.0.0.0
7: gretap0@NONE: <BROADCAST,MULTICAST,NOARP> mtu 1462 qdisc noop state DOWN group default qlen 1
   link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
8: ip_vti0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1
   link/lp/lp 0.0.0.0 brd 0.0.0.0
9: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1
   link/att 0.0.0.0 brd 0.0.0.0
10: ip6tnl0@NONE: <NOARP> mtu 1452 qdisc noop state DOWN group default qlen 1
   link/tunnel6 :: brd ::
[VM2 out] #
```

Packet Capture (Middle Panel):

No.	Time	Source	Destination	Protocol	Length	Info
392	146.012967455	100.64.0.3	100.64.0.2	WireGuard	178	Transport Data, receive
393	147.000523620	100.64.0.2	100.64.0.3	WireGuard	178	Transport Data, receive
394	147.011942952	100.64.0.3	100.64.0.2	WireGuard	178	Transport Data, receive
395	148.000413874	100.64.0.2	100.64.0.3	WireGuard	178	Transport Data, receive
396	148.011780391	100.64.0.3	100.64.0.2	WireGuard	178	Transport Data, receive
397	149.000591643	100.64.0.2	100.64.0.3	WireGuard	178	Transport Data, receive
398	149.012851117	100.64.0.3	100.64.0.2	WireGuard	178	Transport Data, receive
399	150.007321487	100.64.0.2	100.64.0.3	WireGuard	178	Transport Data, receive
400	150.013834601	100.64.0.3	100.64.0.2	WireGuard	178	Transport Data, receive
401	151.000725335	100.64.0.2	100.64.0.3	WireGuard	178	Transport Data, receive
402	151.012445612	100.64.0.3	100.64.0.2	WireGuard	178	Transport Data, receive

Terminal Window (Right Panel):

```
valid_lft forever preferred_lft forever
3: ipddp0: <BROADCAST,MULTICAST,NOARP> mtu 585 qdisc noop state DOWN group default qlen 1000
   link/lp/ddp 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
4: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default q0
   link/ether 00:00:00:00:00:02 brd ff:ff:ff:ff:ff:ff
   inet 100.64.0.2/24 scope global eth1
   valid_lft forever preferred_lft forever
   inet6 fe80::200:ff:fe00:2/64 scope link
   valid_lft forever preferred_lft forever
5: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1
   link/lp/lp 0.0.0.0 brd 0.0.0.0
```

Figura 11: Ping entre PCs.

Implementación en hardware

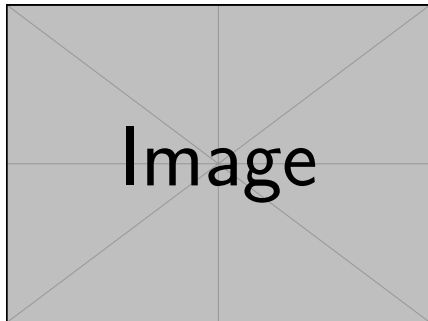


Figura 12: SuperMicro SYS-E300-9D.

Implementación en hardware

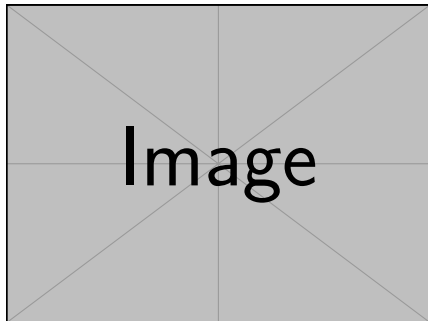


Figura 12: SuperMicro SYS-E300-9D.

Desafíos:

- ✓ Redirección de consola.
- ✗ Passthrough de controlador Ethernet.
- ✗ Throughput entre VMs.

① Introducción

② Revisión bibliográfica

③ Desarrollo

④ Conclusiones

Conclusiones

- Algo

¡Muchas gracias!
¿Preguntas?