PROYECTO INTEGRADOR DE LA CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

MI SÚPER INTERESANTE PI

Juan Teleco
Estudiante

Ing. Pepe Antenil
Director

Miembros del Jurado
Señor Malo 1 (INVAP)
Señora Mala 2 (Instituto Balseiro)

4 de Junio de 2025

Telecolandia

Instituto Balseiro
Universidad Nacional de Cuyo
Comisión Nacional de Energía Atómica
Argentina

A todos los telequitos

Índice de símbolos

Índice de contenidos

Índice de símbolos v

Índice de contenidos vii

Índice de figuras ix

Índice de tablas xi

Resumen xiii

Abstract xv

1. Introducción 1

1.1. Motivación . . . 1
1.2. Concepto de operaciones . . . 1
1.2.1. Antecedentes . . . 1
1.2.2. Suposiciones y restricciones . . . 1
1.2.3. Resumen del sistema propuesto . . . 1
1.2.4. Objetivos, metas y justificación del sistema . . . 1
1.2.5. Usuarios y modos de operación . . . 1
1.3. Requerimientos . . . 1
1.3.1. Funcionales . . . 1
1.3.2. De rendimiento . . . 1
1.3.3. De interfaz . . . 1

2. Arquitectura propuesta 3

2.1. Descripción de tecnologías . . . 3
2.1.1. WireGuard . . . 3
2.1.2. seL4 . . . 3
2.1.3. CAmkES . . . 3
2.2. Arquitectura lógica . . . 4
2.2.1. Dominios . . . 4

3. Estrategia de modelos en entornos virtualizados 5

3.1. Sistema mínimo . . . 5
3.2. Sistema completo usando máquinas virtuales . . . 6
3.3. Sistema completo sobre seL4 . . . 6
3.4. Resumen . . . 6

4. Implementación en entorno virtualizado **7**
4.1. Diseño del experimento . . . 7
4.2. Procedimiento . . . 7
4.2.1. Construcción de un kernel Linux con soporte para WireGuard . . . 7
4.2.2. Generación de una imagen de sistema mediante Buildroot . . . 7
4.2.3. Adaptación del ejemplo *zmq_samples* . . . 7
4.2.4. Configuración del *passthrough* de interfaz Ethernet . . . 8
4.3. Integración . . . 8
4.4. Validación . . . 8

5. Implementación en hardware **9**
5.1. SuperMicro SYS-E300-9D . . . 9
5.2. Diseño del experimento . . . 9
5.3. Procedimiento . . . 9

A. Apéndice I **11**

Bibliografía **13**

Publicaciones asociadas **15**

Agradecimientos **17**

Índice de figuras

2.1. VMM. 3
2.2. Esquema de comunicación zmq_samples. 4

3.1. Otro esquema mejorcito. 5
3.2. Otro esquema mejorcito. 6

Índice de tablas

Resumen

Este es el resumen en castellano.
La tesis debe reflejar el trabajo desarrollado, mostrando la metodología utilizada, los resultados obtenidos y las conclusiones que pueden inferirse de dichos resultados.

Palabras clave: FORMATO DE TESIS, LINEAMIENTOS DE ESCRITURA, INSTITUTO BALSEIRO

Abstract

This is the title in English:
The thesis must reflect the work of the student, including the chosen methodology, the results and the conclusions that those results allow us to draw.

Keywords: THESIS FORMAT, TEMPLATES, INSTITUTO BALSEIRO

Capítulo 1

Introducción

1.1. Motivación

1.2. Concepto de operaciones

1.2.1. Antecedentes

1.2.2. Suposiciones y restricciones

1.2.3. Resumen del sistema propuesto

1.2.4. Objetivos, metas y justificación del sistema

1.2.5. Usuarios y modos de operación

1.3. Requerimientos

1.3.1. Funcionales

1.3.2. De rendimiento

1.3.3. De interfaz

Capítulo 2

Arquitectura propuesta

2.1. Descripción de tecnologías

2.1.1. WireGuard

Ejemplo de un archivo de configuración wg0.conf

2.1.2. seL4

2.1.3. CAmkES

Trabajar con un microkernel como seL4 implica que la mayoría de las funcionalidades del sistema deben implementarse en el espacio de usuario. Para simplificar este proceso, seL4 incluye CAmkES, un *framework* que facilita el desarrollo de sistemas de software modulares y seguros, diseñados específicamente para ejecutarse sobre seL4.

Modular refiere a que CAmkES es un modelo de componentes. Seguro refiere a. Detallar el componente VMM, será utilizado después.

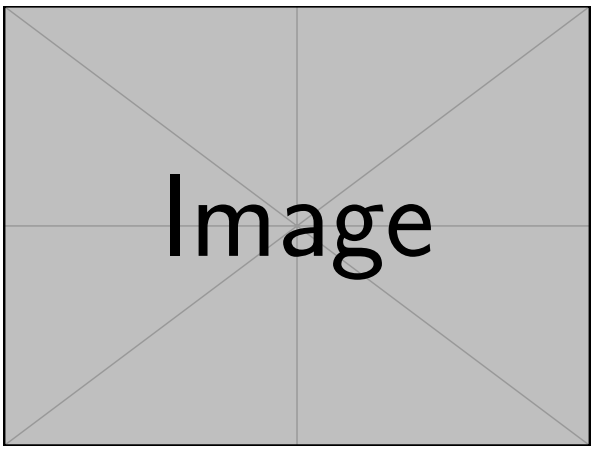

Figura 2.1: VMM.

VirtQueue

Comunicación VM - VMM - seL4 - Hardware

zmq_samples

Uno de los ejemplos de uso de VMs en CAmkES es el proyecto zmq_samples, que implementa un sistema de comunicación entre VMs utilizando la librería de mensajería ZeroMQ. Cada VM contiene una interfaz de red virtual eth0 que se conecta a las interfaces eth0 de las demás. En la figura 2.2 se esquematiza el funcionamiento de este sistema.

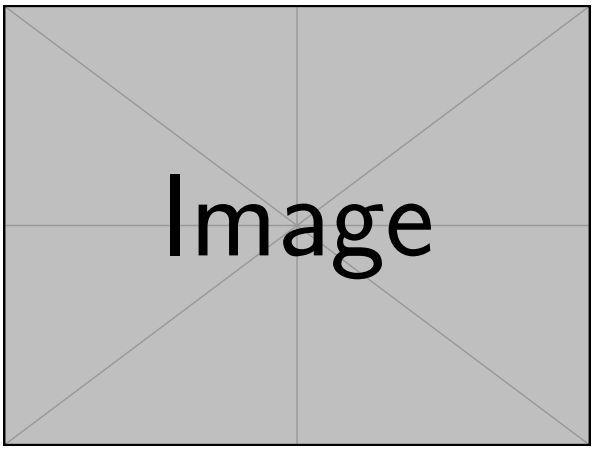

Figura 2.2: Esquema de comunicación zmq_samples.

2.2. Arquitectura lógica

2.2.1. Dominios

Capítulo 3

Estrategia de modelos en entornos virtualizados

Con el fin de abordar de manera ordenada la complejidad del sistema y validar progresivamente cada uno de sus componentes, se adoptó una estrategia basada en la construcción de modelos incrementales. Estos modelos permiten simular, probar y verificar distintas funcionalidades antes de integrarlas en la solución final.

Los modelos descritos a continuación tienen como finalidad:

- Ligar problemas concretos a cada modelo y resolverlos de forma independiente.
- Obtener una solución funcional en un entorno virtualizado como último paso previo a probarla sobre *hardware* real.

3.1. Sistema mínimo

Se propuso un primer modelo de baja complejidad complementario a la lectura de la documentación de WireGuard. La finalidad de este es familiarizarse con la configuración de un túnel VPN y su funcionamiento a través del análisis de paquetes de red. En el esquema de la figura 3.1 se representa la topología de este ensayo, implementada sobre GNS3.

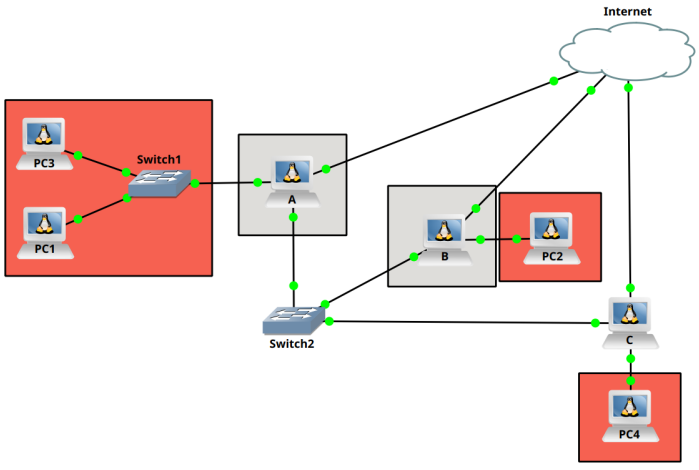

Figura 3.1: Otro esquema mejorcito.

En una siguiente iteración de este modelo se descartó la utilización de GNS3 y se optó por instanciar

cada VM en QEMU, y de esta manera adquirir mayor control sobre los dispositivos de red emulados. Un ejemplo de esto es la posibilidad de observar los parámetros PCI de las interfaces de red, lo cual es relevante para la implementación del *passthrough* de las mismas en la solución final.

Sobre este modelo además puede validarse el correcto funcionamiento de un kernel Linux modificado que luego se utilizará en cada VMM de seL4.

3.2. Sistema completo usando máquinas virtuales

Una vez planteada la arquitectura lógica del sistema, se procede a su simulación en GNS3. Este modelo permite limitar la complejidad de implementar la arquitectura a configurar la interconexión de las VMs que conforman el encriptador.

Este modelo tiene como objetivo validar la arquitectura lógica propuesta y verificar las funcionalidades de red pretendidas para el encriptador como puede ser el *split-tunneling*.

En la figura 3.2 se muestra la topología de red utilizada en GNS3 para simular el sistema completo. El encriptador se implementa como la interconexión mediante interfaces de red de tres VMs independientes.

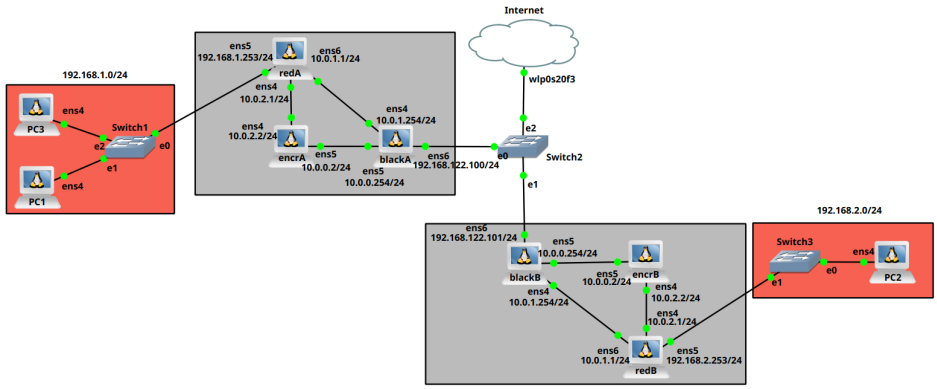

Figura 3.2: Otro esquema mejorcito.

Como *output* de este modelo se obtienen las configuraciones de red necesarias (tablas de enrutamiento, reglas de firewall, etc.) para darle funcionalidad al encriptador. Estas configuraciones se utilizarán posteriormente en la solución.

3.3. Sistema completo sobre seL4

- 4 instancias de QEMU: 2 PCs y 2 encriptadores.
- Descripción de zmq_samples. Virtual Switch. ZeroMQ. Esquemas.

3.4. Resumen

Tabla: modelo, nivel de complejidad, validaciones

Capítulo 4

Implementación en entorno virtualizado

4.1. Diseño del experimento

- Se realizará en completamente en host. ¿Por qué? Simplicidad de implementación.

4.2. Procedimiento

4.2.1. Construcción de un kernel Linux con soporte para WireGuard

- De www.wireguard.com/compilation/kernel-requirements se obtuvieron los requerimientos.
- wireguard-linux-compat patch
- seL4 soporta kernel 4.9[ref], se compiló este kernel con una adaptación de .config file original de los ejemplos camkes-vm
- Buscar referencias para justificar todo. [1]
- Drivers de red compatibles con el hardware utilizado.

4.2.2. Generación de una imagen de sistema mediante Buildroot

- Se utilizó la versión 2023.02.1 de Buildroot utilizando el kernel modificado.
- Se configuró el sistema de archivos para que contenga los binarios necesarios para el funcionamiento de WireGuard y las herramientas de red.

4.2.3. Adaptación del ejemplo *zmq_samples*

- Se adaptó el ejemplo de CAmkES para que funcione con el nuevo kernel y la imagen de sistema generada. Basicamente tocar el CMakeLists.txt
- ZeroMQ. Problema con iperf3 solucionado aumentando tamaño de buffers.
- Incrementar RAM de las VMs.

4.2.4. Configuración del *passthrough* de interfaz Ethernet

- Interfaz e1000 QEMU. PCI. BARS. IRQ.
- Modificar la configuración de la VMM camkes en seL4 para permitir el acceso a los recursos PCI correspondientes.
- Solución al problema de utilizar dos interfaces de red. Diferentes IRQ.
- Funcionamiento de passthrough (colas).

4.3. Integración

- Bridge en host. 4 instancias de QEMU. 2 PCs y 2 encriptadores.

4.4. Validación

Capítulo 5

Implementación en hardware

5.1. SuperMicro SYS-E300-9D

- Supermicro server sys-e300-9d with X11SDV-4C-TLN2F motherboard and intel xeon D-2123IT. Fotito del equipo.
- IPMI, COM1, SoL.

5.2. Diseño del experimento

- Que cosas esperamos validar. Rendimiento y que más?
- SetUP.

5.3. Procedimiento

- 1. Configurar redirección consola serie.
- 2. Bootear el Linux para obtener los parámetros para configurar el zmq_samples (lspci).
-

Apéndice A

Apéndice I

Bibliografía

[1] Laricchia, G., Armitage, S., Köver, A., Murtagh, D. J. Ionizing collisions by positrons and positronium impact on the inert atoms. En: Advances in Atomic, Molecular and Optical Physics, tomo 56, pág. 3. Academic Press, Inc., 2009. 7

Publicaciones asociadas

1. Mi primer aviso en la revista **ABC**, 1996
2. Mi segunda publicación en la revista **ABC**, 1997

Agradecimientos

A todos los que se lo merecen, por merecerlo