Índice de contenidos

Capítulo 1

Introducción

1.1. Motivación y objetivos

En el ámbito de las comunicaciones, es fundamental contar con un sistema capaz de garantizar confiabilidad respecto a la autenticación, confidencialidad e integridad de la información transmitida. Cuando la comunicación se desarrolla sobre un medio considerado inseguro, asegurar el cumplimiento de estas propiedades implica abordar desafíos, como mitigar la posibilidad de suplantación de identidad, encriptar la información para asegurar confidencialidad y detectar modificaciones del mensaje por agentes externos.

Se han desarrollado soluciones privativas al problema descrito. Sin embargo, la inexistencia de *back-doors* y mecanismos de vigilancia es algo que no puede ser verificado por completo en dichas soluciones. Es por esto que en áreas como defensa y servicio diplomático es de interés contar con equipos completamente auditables, que no dependan de licencias de exportación. A su vez, que la funcionalidad y validación formal del equipo no dependa de las funciones criptográficas utilizadas, y que estas puedan ser provistas o implementadas por el usuario final.

El continuo aumento del poder de cómputo ha permitido la ejecución de múltiples sistemas operativos sobre un mismo *hardware*. Esto habilitó la adopción de tecnologías de virtualización en sistemas embebidos y de misión crítica. Una de las principales motivaciones detrás de esta tendencia es la capacidad de establecer fronteras fuertes entre dominios con distintos niveles de confianza, como es el caso de los dominios rojo/negro en sistemas de comunicaciones.

En este trabajo se propone un abordaje novedoso a las soluciones de encriptación de redes: la utilización de tecnologías de virtualización para la segmentación de dominios rojo/negro en un encriptador. Los objetivos que se pretenden alcanzar se resumen en los siguientes:

- Validar la viabilidad de realizar segmentación virtual de dominios.
- Realizar una prueba de concepto del enfoque propuesto.
- Implementar una propuesta de solución auditable y documentada.

La continuación de este capítulo tiene como propósitos: (i) brindar una visión general sobre conceptos relacionados a sistemas de comunicaciones seguras, (ii) presentar el sistema propuesto y su justificación y (iii) describir los requerimientos para con el sistema a desarrollar.

1.2. Concepto de operaciones

El CONOPS es un documento utilizado para describir cómo se espera que opere un sistema desde el punto de vista del usuario final, sin brindar aún detalles técnicos de la solución. Esta sección es de utilidad para clarificar el propósito y uso del sistema, y sirve además de base para la definición de requerimientos.

1.2.1. Contexto

La tríada de la CIA *(Confidentiality, Integrity, Availability)* es un modelo que constituye la base para el desarrollo de sistemas de seguridad. Es utilizada para identificar vulnerabilidades de un sistema y proponer soluciones que cumplan con estos principios.

- **Confidencialidad**: consiste en proteger la información sensible de accesos no autorizados. Los métodos para reforzar este aspecto pueden involucrar la encriptación de la información y la implementación de controles de acceso.

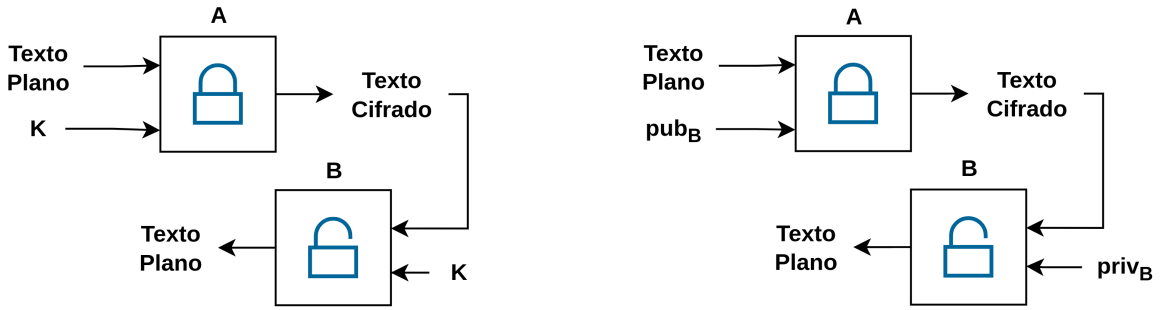
- **Integridad**: refiere a asegurar la consistencia y confiabilidad de la información transmitida y mitigar el riesgo de que los mensajes sufran alteraciones por parte de agentes no autorizados. La implementación de firmas digitales es uno de los métodos empleados para reforzar este principio.

- **Disponibilidad**: cumplir con este aspecto requiere asegurar que la información sea accesible para usuarios autorizados cada vez que sea requerida. Un sistema robusto en este aspecto tiene que ser capaz de soportar ataques de denegación de servicio. Los métodos que refuerzan este aspecto pueden involucrar introducir redundancia a los componentes del sistema.

Un canal inseguro es un medio de transmisión en el que la información se encuentra expuesta a ataques. La escucha pasiva, la suplantación de identidad y la denegación de servicio son algunos ejemplos de ataques típicos de un canal inseguro. Los primeros dos corresponden al área de confidencialidad en el modelo CIA, y pueden tratarse implementando métodos de encriptación. La encriptación de las comunicaciones garantiza la confidencialidad de las mismas y es la base de un sistema de comunicaciones seguras. Existen dos enfoques principales para realizar encriptación, el enfoque simétrico y el asimétrico:

- **Encriptación simétrica**: consiste en un método en el cual las partes utilizan una misma clave para la encriptación y desencriptación de la información. Esto trae como problema que cualquier entidad con acceso a dicha clave tiene la capacidad de leer y reescribir la información. Aún así, se trata de un método eficiente y muy utilizado para transmitir grandes volúmenes de datos. Cualquier implementación con este enfoque requiere como complemento de una forma segura de intercambiar la clave utilizada en un contexto de canales inseguros debido a que mantener la confidencialidad de la comunicación depende de que ambas partes resguarden la clave simétrica.

- **Encriptación asimétrica**: bajo este enfoque cada entidad posee un par de claves únicas, denominadas clave pública y clave privada, que guardan relación entre sí. La clave pública es utilizada para encriptar información y la clave privada correspondiente al mismo par es utilizada para desencriptarla. Este método no suele ser utilizado para la transmisión de grandes volúmenes de información debido a que, al ser de mayor complejidad computacional, se vuelve poco eficiente en estos casos. A diferencia del enfoque simétrico, aquí la confidencialidad de lo que transmite una de las partes depende de que la misma mantenga asegurada su clave privada.



(a) Encriptación simétrica **(b)** Encriptación asimétrica

Figura 1.1: Comparación entre encriptación simétrica y asimétrica.

El método de Diffie-Hellmann propone combinar ambos enfoques para lograr acordar, empleando el concepto de encriptación asimétrica, una clave simétrica de manera segura que puede ser usada posteriormente para encriptar información. La principal ventaja del método radica en que ambas partes logran generar la misma clave simétrica sin transmitirla por el canal, mitigando un gran problema de seguridad del enfoque simétrico. La figura 1.2 describe las operaciones matemáticas sobre las cuales funciona el método de Diffie-Hellmann.

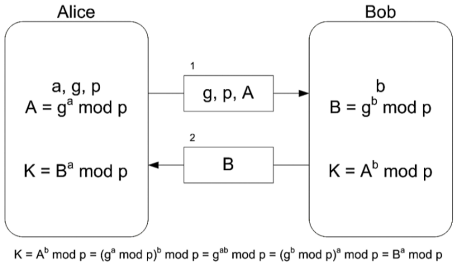


Figura 1.2: Método de Diffie-Hellmann para el acuerdo de una clave común.

Para los estándares de seguridad actuales, un esquema de establecimiento de comunicaciones requiere mayor complejidad para abordar desafíos como un ataque de tipo man-in-the-middle. Aún así, el método de Diffie-Hellmann es la base de numerosas implementaciones por su simplicidad y el elevado costo computacional que supone calcular la clave K conociendo únicamente las claves públicas A y B.

1.2.2. Suposiciones y restricciones

Distintas soluciones de encriptación pueden proponerse según el nivel del modelo de capas que se analice. La encriptación a nivel de capa física minimiza las penalidades en rendimiento a costa de introducir complejidad al sistema, que se manifiesta en la necesidad de hardware dedicado de extremo a extremo de la red. Esta restricción puede volver inviable la encriptación a nivel de la capa física cuando el sistema de comunicaciones requiere de escalabilidad, principalmente por el costo de despliegue y mantenimiento. La encriptación en capas superiores provee mayor flexibilidad en la implementación del sistema. Si bien esto introduce una mayor latencia a la red, reduciendo el rendimiento de la misma, se pueden lograr valores aceptables de latencia con suficiente optimización. Las soluciones de encriptación en la capa de red tienen la ventaja de ser independientes de la capa física, reduciendo la complejidad del sistema y permitiendo mayor flexibilidad en la implementación y compatibilidad con infraestructura preexistente. Esto implica que, cualquier sitio que cuente con una conexión a Internet

y un dispositivo de encriptación tiene la infraestructura suficiente para acceder a una red segura, denominada red privada virtual.

En sistemas de criptografía, usualmente se utiliza el concepto de dominios rojo/negro para describir las partes del sistema que trabajan con información legible (dominio rojo) y aquellas que contienen información cifrada (dominio negro). La arquitectura de un sistema de comunicación seguro debe tener en cuenta este concepto para una correcta segregación de dominios, mitigando así la posibilidad de filtraciones indeseadas de información. La normativa actual que refiere a la segregación de dominios no contempla la segregación virtual, es decir, contener en un mismo dispositivo ambos dominios y aislarlos empleando herramientas de software. El implementar un sistema de este estilo capaz de cumplir las normas de seguridad existentes es un desafío importante.

1.2.3. Resumen del sistema propuesto

El sistema de comunicaciones seguras que se propone contempla el diseño de un único dispositivo, denominado encriptador, con la capacidad de formar un túnel VPN entre redes preexistentes, asegurando ciertos estándares de autenticación y confidencialidad sobre las comunicaciones entre dichas redes. El sistema opera a nivel de capa de red, y requiere de un mínimo de dos encriptadores para su funcionamiento, aunque también se encuentra prevista la escalabilidad del sistema. El propósito del sistema es brindar confiabilidad respecto a la autenticación, confidencialidad e integridad de la información transmitida entre redes interconectadas, afectando al mínimo el rendimiento de la red.

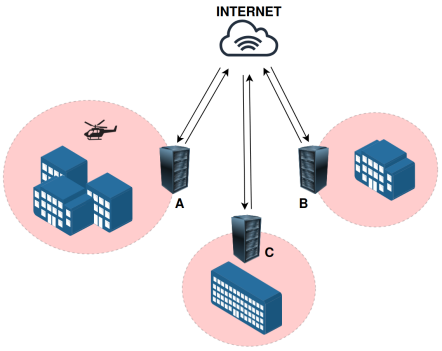


Figura 1.3: Esquema simplificado del sistema de comunicaciones seguras.

En el esquema de la figura 1.3 se describen los denominados dominios rojos, correspondientes a redes donde se trata con información sensible. El complemento de estos se denominan dominios negros, que usualmente se consideran canales inseguros donde la información proveniente de un dominio rojo requiere estar encriptada. El dispositivo encriptador actúa de interfaz entre dominios rojo y negro, motivo por el cual gran parte de la seguridad del sistema recae sobre este. Este dispositivo cuenta con acceso a claves utilizadas para establecer sesiones seguras con otros dispositivos, y al mismo tiempo cuenta con acceso a Internet. Debe poder garantizarse la seguridad en la gestión de estas claves y rechazar accesos no autorizados al dominio rojo.

1.2.4. Objetivos, metas y justificación del sistema

Un dispositivo en el cual los dominios rojo y negro no estén correctamente segmentados se encuentra con que la información confidencial respecto al túnel VPN como claves y permisos se hallan expuestos en el caso de una intrusión desde Internet o desde la propia red interna. Esto lleva a la posibilidad de que un agente no autorizado sea capaz de desencriptar y modificar información que viaje por el túnel, así como también acceder a dispositivos dentro de una organización.

En el diseño del encriptador, se propone implementar la segmentación virtual de dominios rojo/-negro, esto es, representar los dominios como entidades virtuales independientes que ejecutan sobre hardware compartido. Este concepto puede implementarse a través de los denominados hipervisores, software que permite que varios sistemas operativos independientes trabajen juntos, compartiendo los mismos recursos físicos.

La segmentación permite que la entidad negra, la cual está conectada a Internet, realice el control del tráfico y oculte la existencia de la entidad roja para cualquier servicio fuera de la red virtual privada. Por otro lado, la entidad roja es la responsable de encriptar/desencriptar las comunicaciones entre organizaciones que se encuentren dentro de la red privada y de gestionar las claves de encriptación y los privilegios de usuarios. Esta segmentación permite un control estricto sobre el tráfico entre ambas entidades, mitigando la posibilidad de transferir información como claves contenidas en la entidad roja hacia la entidad negra.

1.2.5. Usuarios y modos de operación

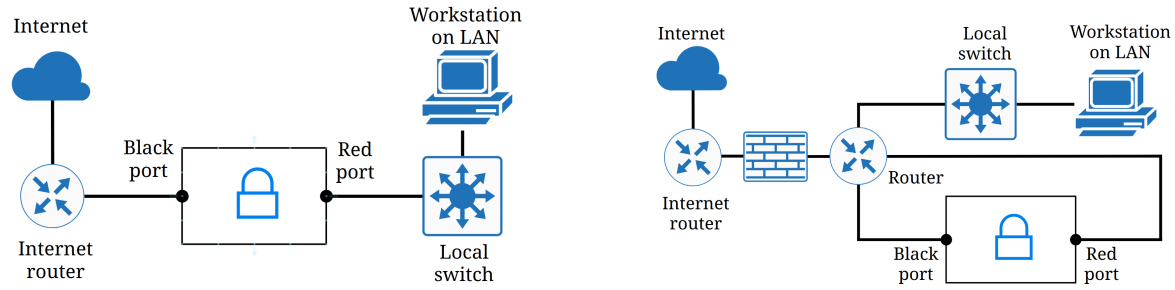
Usuarios

- **Administrador de red**: responsable de la configuración y mantenimiento del encriptador. Cuenta con acceso a la interfaz de administración del dispositivo y es el encargado de configurar el encriptador para funcionar dentro de la red segura. La configuración del encriptador incluye la definición de las redes que se conectarán a través del túnel VPN, la configuración de las claves de encriptación y la definición de los permisos de los usuarios.
- **Usuario final**: utiliza la red segura para compartir información sensible a otros nodos de la red. No tiene acceso a la configuración del encriptador y su interacción con el sistema se limita a utilizar la red segura como si de su red local se tratase.

Modos de operación

El diseño prevee que, una vez configurado el dispositivo, la operación del encriptador sea transparente para el usuario final, el cual no necesita interactuar con el dispositivo para utilizar la red segura.

- **Nodo pequeño:** este modo de operación es de utilidad en sitios que no cuentan con infraestructura de red. El encriptador opera también como router, procesando todo el tráfico del sitio y permitiendo que el tráfico no destinado a otro sitio de la red segura sea enrutado por fuera del túnel VPN. Esta funcionalidad es conocida como *split tunneling*. Es adecuado para instalaciones con tráfico moderado.
- **Nodo grande:** en instalaciones con tráfico elevado y que posiblemente cuenten con una infraestructura de red acorde, el equipo únicamente opera como encriptador de las comunicaciones seguras, y se conecta a un router preexistente para las comunicaciones no seguras. De esta manera solamente el tráfico dentro de la red segura es procesado por el encriptador.



(a) Sitio sin infraestructura de red.

(b) Sitio con infraestructura de red.

Figura 1.4: Modos de operación del encriptador.

1.3. Requerimientos

El documento de requerimientos es utilizado para describir que necesita el sistema para cumplir las necesidades de los usuarios. Esta sección establece las bases para la solución propuesta. Se definen a continuación las palabras clave que se utilizarán para referirse a los requerimientos:

Palabra clave	Descripción
DEBE	Indica un requerimiento obligatorio que debe cumplirse para que el sistema funcione correctamente.
NO DEBE	Indica un requerimiento que no debe cumplirse, es decir, una restricción que debe evitarse.
DEBERÍA	Indica un requerimiento recomendado, pero no obligatorio. Se sugiere cumplirlo para mejorar la calidad del sistema.
NO DEBERÍA	Indica un requerimiento que se desaconseja cumplir, pero no es obligatorio evitarlo.
PUEDE	Indica una opción o característica que el sistema puede implementar, pero no es obligatoria.

Tabla 1.1: Palabras clave utilizadas en la definición de requerimientos.

1.3.1. Funcionales

- **Renovación de claves:** cada par de nodos realiza una renovación de claves efímeras cada 120 segundos para asegurar forward-secrecy. Esto significa que, si una clave es comprometida por alguna razón, no se comprometen las comunicaciones anteriores o futuras fuera del intervalo de tiempo especificado.

- **Seguridad ante intrusiones:** el equipo debe ser capaz de mitigar la posibilidad de intrusiones de agentes no autorizados vía software, tanto desde Internet o como desde la red local.

- **Detección de ataques DoS:** el sistema debe ser capaz de detectar y mitigar ataques de denegación de servicio.

- **Movilidad:** el sistema debe ser capaz de soportar la movilidad de los nodos y permitir a un nodo moverse entre redes sin interrupciones ni renegociaciones de claves.

- **Split-tunneling:** el sistema debe ser capaz de permitir y enrutar tráfico de ciertas aplicaciones o servicios por fuera del túnel VPN.

- **Segmentación de dominios:** el sistema debe aislar procesos contenidos en un dominio de información, como pueden ser archivos, contenida en otro dominio, independientemente de los privilegios que tenga este proceso.

1.3.2. De rendimiento

- **Tasa de transferencia:** el sistema debe ser capaz de lograr, de un nodo a otro, una tasa de transferencia de 950 Mbits/s de datos planos.
- **Número de nodos:** una red segura debe ser capaz de soportar hasta 250 dispositivos encriptadores, también denominados nodos.

1.3.3. De interfaz

- **Administración:** la configuración de funcionamiento del encriptador debe poder ser modificada únicamente por un administrador de red autorizado de manera local.
- **Interfaz de usuario:** el sistema debe ser transparente para el usuario final, es decir, no debe requerir de configuraciones adicionales para este.
- **Configuración de operación:** el modo de operación en red del sistema y otros parámetros de funcionamiento asociados deben ser configurables únicamente por el administrador de red.

1.4. Descripción de tecnologías

En este trabajo se combinarán distintas tecnologías que conformarán la solución final. A continuación se describen brevemente a modo de contextualizar al lector.

1.4.1. WireGuard

WireGuard es un protocolo de túneles VPN derivado de Noise Protocol Framework. Está diseñado para ser simple, rápido y eficiente, proporcionando una solución de código abierto con una base de código pequeña, lo que ayuda a minimizar la superficie de ataque. Su propósito es proporcionar una alternativa segura y eficiente a los protocolos VPN tradicionales, permitiendo la creación de túneles seguros entre dispositivos en una red.

1.4.2. seL4

Se trata de un microkernel con verificación formal matemática que garantiza que su implementación está libre de errores. Cuenta con una base de código muy reducida y proporciona un fuerte aislamiento entre procesos y una interfaz de comunicación segura. Es utilizado en sistemas críticos donde la seguridad y la confiabilidad son fundamentales.

1.4.3. 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.

Capítulo 2

Arquitectura propuesta

Este capítulo tiene como objetivo introducir la propuesta de solución planteada para el encriptador.

2.1. Arquitectura lógica

Un hipervisor es un *software* que permite la creación y ejecución de múltiples máquinas virtuales, las cuales se ejecutan sobre *hardware* compartido de manera aislada e independiente. El hipervisor gestiona los recursos de *hardware* para cada máquina virtual según sea necesario, además provee una interfaz de comunicación entre las máquinas virtuales y el *hardware*.

El diseño propuesto está basado en la utilización de un hipervisor, el cual administra tres máquinas virtuales denominados dominios. En la figura 2.1 se muestra un esquema de la arquitectura propuesta donde se presenta tanto la lógica interna del dispositivo a través de la interconexión de los dominios como la interacción con la red externa al encriptador.

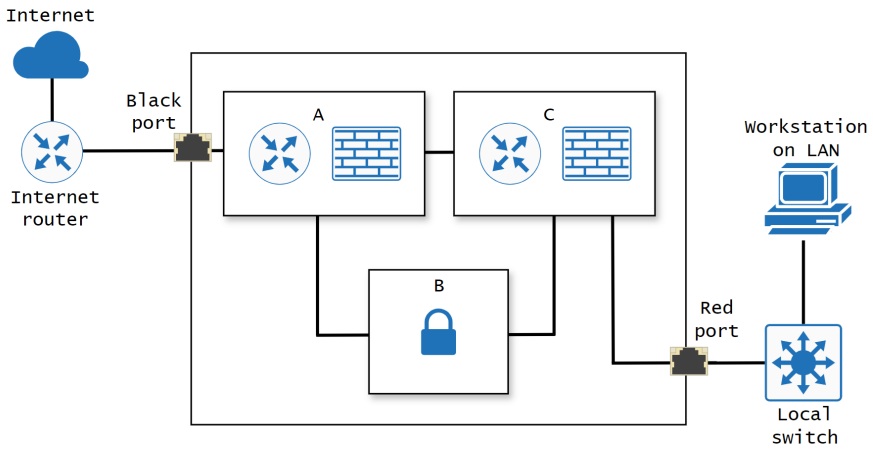


Figura 2.1: Esquema de arquitectura lógica del encriptador.

2.1.1. Dominio A - Negro

El dominio A cuenta con tres interfaces de red, una física que lo conecta con el puerto de internet y dos virtuales que lo conectan con los dominios B y C. Esta entidad actúa como primer firewall para el tráfico proveniente de internet y direcciona el tráfico hacia B o C según la dirección IP de origen de los paquetes. Si el tráfico proviene de otro sitio dentro de la red segura, este se direcciona hacia

B para ser desencriptado. En el caso de ser tráfico proveniente de cualquier otro servicio de internet, este se direcciona hacia el dominio C.

2.1.2. Dominio B - Encriptador

Este dominio cuenta únicamente con dos interfaces virtuales, las cuales conectan a este dominio con A y C. Esta entidad administra el software necesario para el funcionamiento del túnel VPN, gestiona las claves para la conexión con otros nodos y realiza la encriptación/desencriptación de los paquetes salientes/entrantes respectivamente. Como base para la realización de estas funciones la entidad utiliza el software WireGuard. La implementación de Wireguard requiere almacenar un par de claves pública y privada propios de cada nodo, además de la clave pública de demás nodos de la red. Estas claves son almacenadas en un archivo de configuración encriptado dentro de este dominio.

2.1.3. Dominio C - Rojo

Este dominio cuenta con una interfaz de red virtual que lo conecta con el dominio A y una interfaz física que lo conecta a la la red local del sitio. Esta entidad actúa como un segundo firewall para el tráfico proveniente de internet y direcciona el tráfico saliente hacia A o B según la dirección IP de destino de los paquetes provenientes de la red local. Si el tráfico tiene como destino un servicio de internet, este se direcciona hacia A, sin encriptación dedicada. En el caso de tener como destino otro sitio dentro de la red segura, este se direcciona hacia B para ser encriptado.

2.2. Arquitectura física

El *hardware* elegido para la implementación del encriptador es el SuperServer E300-9D de Supermicro, entre sus principales características, cuenta con un procesador Intel Xeon D-1521, 32 GB de memoria RAM y dos interfaces de red de 10Gb. Este procesador posee arquitectura x86 y cuenta con soporte para virtualización VT-x y VT-d lo que permite la creación de máquinas virtuales con acceso directo a los dispositivos de *hardware*.

El sistema operativo base del encriptador es seL4 funcionando como hipervisor, este utiliza el VMM *camkes-vm*. A continuación se detalla el funcionamiento e interacción de los componentes descritos hasta el momento.

2.2.1. seL4 hipervisor

2.2.2. vswitch

2.2.3. PCI passthrough

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 implementarla sobre *hardware*.

El detalle de cada modelo tiene como objetivo proporcionar una visión clara de su propósito, las validaciones que se intentan realizar y los aspectos que quedan fuera del alcance de cada uno. Además, en cada modelo se describen aspectos relevantes sobre las tecnologías utilizadas.

3.1. Comunicando sitios seguros con WireGuard

El primer modelo propuesto se centra en la comunicación entre múltiples sitios seguros a través de un túnel VPN configurado mediante WireGuard, permitiendo el acceso de los sitios a otros servicios en Internet por fuera del túnel. Se trata de un modelo de baja complejidad cuyo objetivo principal es familiarizarse con WireGuard y su configuración, así como utilizar herramientas como Wireshark para el análisis de paquetes de red. En una primera instancia, se implementó la topología de la figura 3.1 utilizando GNS3, herramienta que permite instanciar y conectar múltiples máquinas virtuales, integrando herramientas de análisis de redes como Wireshark.

Este modelo no pretende validar la arquitectura lógica del encriptador, sino que se enfoca en la configuración de WireGuard. Es por esto que cada encriptador se representa mediante una sola VM, en la cual se configura WireGuard. Se busca verificar que cada sitio pueda comunicarse con los demás a través del túnel VPN y acceder a Internet de manera directa.

Este modelo permite identificar y resolver problemas básicos de configuración de WireGuard, comprender el funcionamiento de las claves públicas y privadas, y analizar el tráfico cifrado y no cifrado mediante capturas en Wireshark.

En una siguiente iteración de este modelo se descartó la utilización de GNS3 y se optó por instanciar cada VM mediante QEMU, adquiriendo mayor control sobre los dispositivos de red emulados. Un

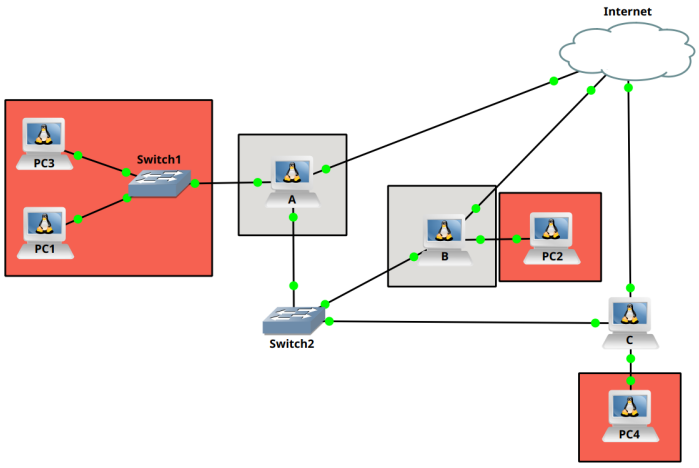


Figura 3.1: Otro esquema mejorcito.

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.

En este modelo, además, se valida el correcto funcionamiento tanto del kernel de Linux como del *filesystem* inicial, modificados para incluir soporte para WireGuard. Estas imágenes serán posteriormente utilizadas en cada VMM gestionada por seL4.

3.2. Introduciendo la arquitectura lógica del encriptador

Una vez planteada la arquitectura lógica del sistema, se procedió con 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. Se tiene como objetivo validar la arquitectura lógica propuesta y verificar las funcionalidades de red pretendidas para el encriptador como lo es el *split-tunneling*.

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

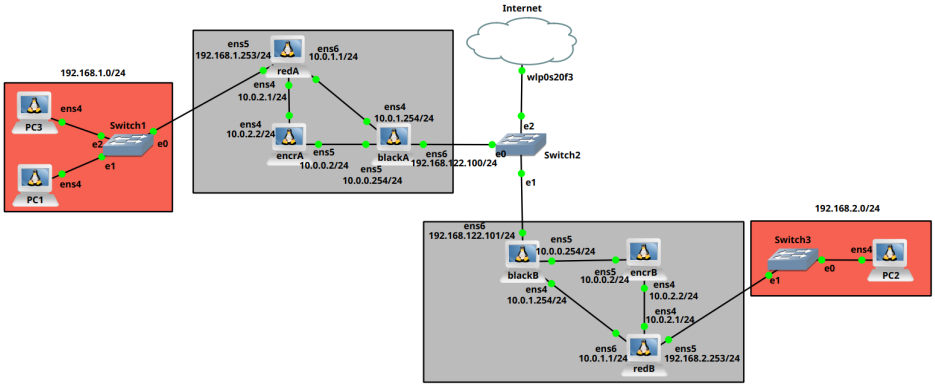


Figura 3.2: Otro esquema mejorcito.

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

3.3. Utilizando seL4 como hipervisor

El siguiente modelo se centra en la lograr la comunicación entre dos VM, las cuales se encuentran en instancias independientes de seL4 funcionando como hipervisor. Este modelo tiene como objetivo validar el *passthrough* de hardware, en este caso, de una interfaz de red entre seL4 y el *guest* Linux. En la figura 3.3 se esquematiza la topología del modelo.

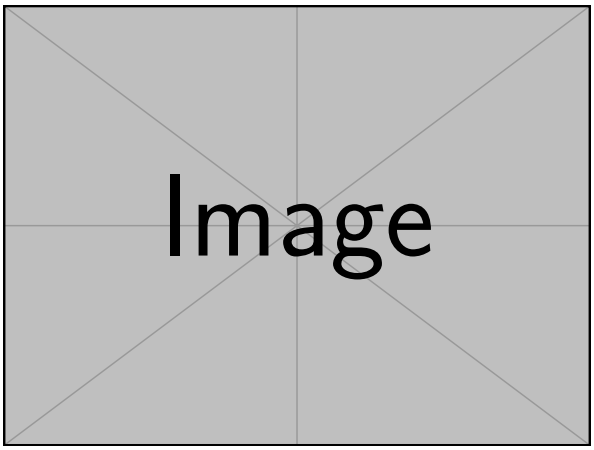


Figura 3.3: Dos VMs con un passthrough, ping 2VM.

Un paso importante de este desarrollo es validar la compatibilidad de las imágenes de Linux obtenidas previamente con el hipervisor seL4. Para ello, en este modelo se utiliza el *output* del primer modelo, la imagen de Linux con soporte para WireGuard.

Como *output* de este modelo se tiene la configuración adecuada para realizar el *passthrough* de un dispositivo de red al Linux *guest*. Además de correcciones que se hayan hecho sobre la imagen de Linux para lograr la compatibilidad con el VMM de seL4.

3.4. Implementando el encriptador en seL4

Este modelo se centra en integrar los desarrollos realizados en los modelos anteriores, y obtener así una solución completa como paso previo a su despliegue sobre *hardware* real. Aquí se validan en conjunto todas las funcionalidades y configuraciones obtenidas previamente, asegurando la interoperabilidad de los distintos componentes en un entorno virtualizado.

Este modelo utiliza uno de los ejemplos de uso de VMs en CAmkES, el proyecto *zmq_samples*, como base para la implementación del encriptador. Este ejemplo permite la comunicación entre dos VMs utilizando ZeroMQ, una biblioteca de mensajería asíncrona que facilita la comunicación entre procesos a través de regiones de memoria compartida definidas. Cada VM contiene una interfaz de red virtual *eth0* que se conecta a las interfaces *eth0* de las demás. En la figura 3.4 se esquematiza el sist completo.

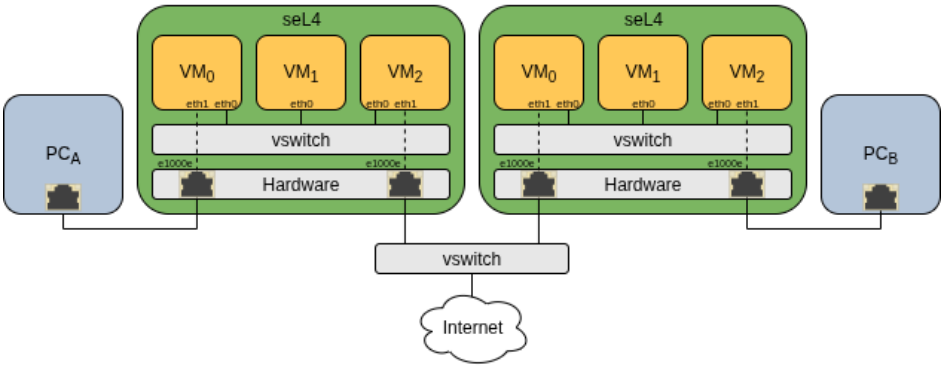


Figura 3.4: Esquema del encriptador con seL4 simulado en QEMU.

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. Generación de una imagen de sistema mediante Buildroot

Buildroot es una herramienta que permite generar imágenes de sistema de archivos o *root filesystem* minimalistas para sistemas embebidos. Es útil para crear sistemas operativos ligeros y personalizados, adaptados a las necesidades específicas de un proyecto. Se utilizó Buildroot para generar una imagen de sistema que contenga los binarios necesarios para el funcionamiento de WireGuard y otras herramientas de red requeridas. En este trabajo se provee el archivo de configuración utilizado para la versión 2023.02.1 de Buildroot y se detalla a continuación el procedimiento para replicar la imagen.

```
1 git clone --branch 2023.11 https://github.com/buildroot/buildroot.git
2 cp .buildroot-config ./buildroot/.config
3 cd buildroot
4 make
```

Código 4.1: Generación de imagen de sistema con Buildroot.

4.2.2. Construcción de un kernel Linux con soporte para WireGuard

Actualmente, el VMM de seL4 implementado en CAmkES para la arquitectura x86 cuenta con soporte probado para el kernel Linux 4.9. En este trabajo se utilizó una versión limpia de este kernel, obtenida de la rama *stable* de Linux 4.9.y, en particular la versión 4.9.337, que se encuentra disponible en el repositorio oficial de Linux. Al momento de configurar la compilación del kernel, se tuvo en cuenta la compatibilidad con el hipervisor seL4, cuyos requisitos se encuentran documentados en la página oficial del proyecto *camkes-vm* para seL4 [**?**].

El soporte nativo para WireGuard fue añadido al kernel Linux en la versión 5.6, sin embargo, el repositorio *wireguard-linux-compat* provee un parche que permite compilar WireGuard en versiones anteriores del kernel, como la utilizada [**?**].

```
git clone https://git.zx2c4.com/wireguard-linux-compat
./wireguard-linux-compat/kernel-tree-scripts/jury-rig.sh ./linux-stable
```

Código 4.2: Parche wireguard-linux-compat para el kernel Linux 4.9.337

4.2.3. Adaptación del proyecto *zmq_samples*

Utilización de kernel modificado

En la configuración por defecto de este proyecto se cuenta con las imágenes de sistema para un kernel Linux 4.8.16 en cada VM. Para utilizar las imágenes de sistema generadas en pasos anteriores se realizaron modificaciones en el archivo *CMakeLists.txt* del proyecto, ubicado en el directorio *camkes-vm-examples-manifest/projects/vm-examples/apps/x86/zmq_samples/*. En este archivo se definieron las rutas de los archivos a utilizar mediante las siguientes variables CMake:

```
set(kernel_file "/host/custom-vm-kernel/linux-stable/arch/x86/boot/bzImage")
set(rootfs_file "/host/custom-vm-kernel/buildroot/output/images/rootfs.cpio")
```

Código 4.3: Variables CMakeLists.txt del proyecto zmq_samples

Gestión de memoria de las VMs

En seL4, toda la memoria física se inicia como *untyped memory*, lo que significa que no está asignada a ningún objeto específico. A medida que se crean objetos, como VMMs, se realizan asignaciones de memoria, convirtiendo parte de la memoria *untyped* en memoria *typed*. Una característica importante de la gestión de memoria en seL4 es que la alocación es estática para objetos en kernel. Esto implica que, al crear una VM, se debe definir la cantidad de memoria que se le asignará desde el inicio. Esta cantidad no puede ser modificada posteriormente, lo que requiere una planificación cuidadosa para evitar problemas de memoria insuficiente durante la ejecución de las VMs.

Cuando se hace uso del VMM *camkes-vm* se tienen los siguientes parámetros de configuración directamente relacionados al uso de memoria de un *guest*:

- `simple_untypedN_pool`: CAmkES realiza una prealocación de memoria *untyped* para ser gestionada por cada VM. Aquí `N` representa una máscara sobre la cantidad de bits que definen el tamaño del bloque de memoria. Por ejemplo, `vm0.simple_untyped23_pool = 10` indica que se reservan 10 bloques de memoria de 2^{23} bytes (8 MB) para el componente `vm0`. Una forma de dar flexibilidad al sistema es definir *pools* de memoria *untyped* de diferentes tamaños.

- `heap_size`: este parámetro define el tamaño del *heap* del componente VMM. Es memoria reservada para el uso del VMM si este requiere por ejemplo guardar información de estado o utilizar buffers de datos. La asignación `vm0.heap_size = 0x40000` define 256 KiB de memoria para el *heap* del componente VMM correspondiente a `vm0`.

- `guest_ram_mb`: la memoria RAM disponible para la VM es simplemente la asignación de memoria *untyped* disponible en *pools* reservados para la misma. Por ejemplo, `vm0.guest_ram_mb = 128` indica que se asignarán 128 MB de memoria RAM a la VM `vm0`. Debe tenerse en cuenta que este valor no puede superar a la cantidad de memoria *untyped* reservada para la VM.

Por defecto se definen...

Configuración del *passthrough* de controlador Ethernet

El hipervisor permite otorgar a una VM acceso directo a un dispositivo de hardware, como una tarjeta de red. Este procedimiento, conocido como *passthrough* de dispositivos PCI, posibilita que la VM utilice el dispositivo como si estuviera conectado físicamente al bus del sistema anfitrión, mejorando el rendimiento en comparación con la emulación del dispositivo.

Un dispositivo PCI se identifica por un número de interrupción (IRQ o MSI) y un conjunto de *Base Address Registers* o BARs que definen las direcciones de memoria y los puertos de entrada/salida que emplea el dispositivo. Estos parámetros deben especificarse en la configuración del VMM para permitir el acceso de la VM a los recursos físicos del dispositivo.

El procedimiento para obtener los parametros necesarios consiste en iniciar un sistema operativo Linux no virtualizado. En este, los dispositivos PCI son detectados y gestionados automáticamente por el sistema. Una vez iniciado, el comando `lspci` permite listar los dispositivos PCI conectados al sistema, proporcionando información detallada sobre cada dispositivo.

BAR	Dirección inicio	Dirección fin	Flags
0	0xfeb80000	0xfeb9ffff	0x00040200
1	0xfeba0000	0xfebbffff	0x00040200
2	0x0000c000	0x0000c01f	0x00040101
3	0xfebd0000	0xfebd3fff	0x00040200
4	0x00000000	0x00000000	0x00000000
5	0x00000000	0x00000000	0x00000000
6	0xfeb00000	0xfeb7ffff	0x00046200

Tabla 4.1: Registros del dispositivo PCI e1000e emulado en QEMU.

En la tabla 4.1 se muestran, a modo de ejemplo, los BARs asociados al dispositivo de red utilizado en este ensayo. El campo de flags permite identificar el tipo de registro, ya sea de memoria o de entrada/salida, determinando que los BAR 0, 1, 3 y 6 son registros de memoria, mientras que el BAR 2 es un registro de entrada/salida. Los BAR 4 y 5 no son utilizados por este dispositivo.

Esta información, en conjunto con el número de IRQ (que en este caso es 11), se utiliza para configurar el VMM de seL4 y permitir que la VM acceda al dispositivo de red. El archivo de configuración del VMM, ubicado en *vm-examples/apps/x86/zmq_samples.camkes*, debe incluir los siguientes campos:

```
vm0.vm_ioports = [
    {"start":0xc040, "end":0xc05f, "pci_device":3, "name":"inner_eth"}
];
vm0.pci_devices = [
    {"name":"inner_eth",
    "bus":0, "dev":0x3, "fun":0, "irq":"inner_eth",
    "memory":[
        {"paddr":0xfeb00000, "size":0x20000, "page_bits":12},
        {"paddr":0xfeb20000, "size":0x20000, "page_bits":12},
        {"paddr":0xfeb90000, "size":0x4000, "page_bits":12}]}
];
vm0.vm_irqs = [
    {"name":"inner_eth", "ioapic":0, "source":0xb, "level_trig":0, "
        active_low":1, "dest":10}
];
```

Código 4.4: Configuración CAmkES para el passthrough del dispositivo de red e1000e.

- 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).
-