## Ciberseguridad Básica

# Resumen de Clase 1

*Autor:*

Santiago Correa Vergara

# Índice

## Licencia



## Nota legal

Este documento ha sido elaborado como un resumen y material complementario basado en la experiencia personal del autor durante un proceso de formación en ciberseguridad. Aunque se ha seguido una estructura temática similar a la de ciertos programas educativos públicos, todo el contenido ha sido reescrito, ampliado y explicado por el autor, sin reproducir directamente ningún material protegido por derechos de autor.

El propósito es estrictamente académico y no comercial.

## Historia y Fundamentos de la Computación

### *Pioneros de la Computación*

**Alan Turing** (1912-1954), considerado el padre teórico de la computación, realizó contribuciones revolucionarias. En 1936 propuso el concepto de la **Máquina de Turing**, un modelo abstracto que formalizó los límites de lo computable. Durante la Segunda Guerra Mundial, su trabajo en Bletchley Park para descifrar el código Enigma salvó incontables vidas. Además, estableció las bases teóricas de los algoritmos y la computabilidad, conceptos que siguen siendo centrales en la informática actual.

**John von Neumann** (1903-1957) desarrolló la arquitectura que lleva su nombre, pilar fundamental de las computadoras modernas. Su diseño introdujo el concepto de **programa almacenado** en memoria, permitiendo que las instrucciones pudieran manipularse como datos. Esta arquitectura organiza la computadora en unidades claramente diferenciadas: la Unidad Central de Proceso (CPU) con su ALU para cálculos aritméticos y lógicos, y la unidad de control para coordinar operaciones.

**Claude Shannon** (1916-2001), con su **teoría de la información** (1948), proporcionó el marco matemático para la era digital. Demostró cómo circuitos con interruptores (precursores de los transistores) podían implementar álgebra booleana, estableciendo así las bases para las operaciones binarias en hardware. Su trabajo conectó directamente la lógica matemática con la implementación física de circuitos digitales.

### *Revolución del Transistor*

El transistor, inventado en 1947 por Bardeen, Brattain y Shockley en Bell Labs, marcó un punto de inflexión en la historia de la computación. Este dispositivo semiconductor, típicamente de silicio o germanio, consta de tres terminales (emisor, base y colector) que permiten controlar el flujo de corriente eléctrica sin partes móviles, actuando como un interruptor electrónico.

Principales características técnicas:

- Funcionamiento basado en propiedades de semiconductores
- Capacidad de conmutación extremadamente rápida (nanosegundos)
- Implementación de puertas lógicas fundamentales (AND, OR, NOT)

Frente a las válvulas termoiónicas que dominaban hasta entonces, los transistores ofrecieron ventajas revolucionarias: un tamaño radicalmente menor (permitiendo mayor miniaturización), una eficiencia energética muy superior, y una mayor velocidad de conmutación. Estas características hicieron posible la explosión de la computación digital moderna.

***Componentes de una Computadora Moderna***

**CPU (Unidad Central de Procesamiento)**: El cerebro del sistema contiene múltiples componentes clave. La **Unidad de Control** dirige el flujo de datos y la ejecución de instrucciones, mientras que la **ALU** (Unidad Aritmético-Lógica) realiza cálculos matemáticos y operaciones lógicas. Los **registros** proporcionan almacenamiento ultrarrápido para datos en procesamiento, y la memoria **caché** (organizada en niveles L1, L2, L3) actúa como buffer entre la CPU y la memoria principal.

**Memoria**: Los sistemas modernos emplean una jerarquía de almacenamiento. La **RAM** (de tipo DRAM o SRAM) ofrece almacenamiento volátil de rápido acceso, mientras la **ROM** almacena firmware crítico como el BIOS. La jerarquía sigue el principio: registros → caché → RAM → almacenamiento persistente.

**Dispositivos de Almacenamiento**: Los **HDD** (discos duros mecánicos) usan platos magnéticos y cabezales móviles, mientras los **SSD** emplean memoria flash NAND para mayor velocidad. Los medios ópticos (CD/DVD/Blu-ray) completan el espectro de opciones de almacenamiento.

**GPU**: Las unidades de procesamiento gráfico destacan por su arquitectura masivamente paralela, con cientos de núcleos optimizados para operaciones matriciales. Originalmente diseñadas para renderizado gráfico, hoy son esenciales en inteligencia artificial y computación científica.

**Placa Madre**: Este componente integra todos los demás mediante su **chipset** (con controladores norte y sur), diversos **buses** de comunicación (PCIe, SATA, USB) y conectores especializados para cada componente.

***Evolución de Arquitecturas***

La historia de las arquitecturas computacionales muestra una progresión constante:

- **Década 1940**: Máquinas electromecánicas como la Z3 alemana o la británica Colossus, diseñada específicamente para descifrar códigos.
- **1950s**: Transición de válvulas termoiónicas a transistores discretos, como en la serie IBM 700.
- **1960s**: Aparición de circuitos integrados que permitieron los mainframes comerciales.
- **1970s**: Revolución del microprocesador con chips como el Intel 4004.
- **1980s**: Explosión de las computadoras personales, liderada por el IBM PC.
- **2000s**: Era del paralelismo con procesadores multinúcleo, GPGPUs, y los primeros pasos en computación cuántica.

## El sistema operativo Libre

### *Richard Stallman: El fundador del movimiento del software libre*

Richard Matthew Stallman (nacido en 1953), conocido como RMS, es un programador estadounidense y fundador del movimiento del software libre. Mientras trabajaba en el laboratorio de IA del MIT en los años 80, tuvo un experiencia crucial cuando no pudieron reparar una impresora Xerox porque el código fuente no estaba disponible. Este incidente lo llevó a crear los **GNU Utils** (1984), las primeras herramientas libres para construir un sistema operativo completo, iniciando así el Proyecto GNU.

Figura 1: Fundador del Proyecto GNU, la Free Software Foundation, GNU utils y defensor del software libre

### *Proyecto GNU, la FSF y la GPL*

El **Proyecto GNU** (1983) surgió con el objetivo de crear un sistema operativo completamente libre. Stallman fundó la **Free Software Foundation** (FSF) en 1985 para apoyar este movimiento. Desarrolló la **Licencia Pública General GNU** (GPL) en 1989, que garantiza cuatro libertades esenciales:

- Libertad 0: Ejecutar el programa como desees
- Libertad 1: Estudiar y modificar el código fuente
- Libertad 2: Redistribuir copias exactas
- Libertad 3: Distribuir versiones modificadas

GNU es un acrónimo recursivo que significa "**GNU's Not Unix**", reflejando su diseño similar a Unix pero siendo software libre.

Figura 2: El proyecto GNU tiene como mascota el Ñu

***GNU Hurd: El kernel original del Proyecto GNU***

GNU Hurd es el kernel desarrollado por el Proyecto GNU como alternativa libre a los kernels Unix. Aunque técnicamente innovador por su diseño micronúcleo, Hurd no alcanzó madurez suficiente para uso general. Características clave:

- **Arquitectura micronúcleo**: Servidores ejecutándose en espacio de usuario para mayor estabilidad
- **Compatibilidad POSIX**: Permite ejecutar aplicaciones Unix estándar
- **Retos técnicos**: Complejidad en la gestión de E/S y drivers
- **Estado actual**: Sigue en desarrollo activo pero no recomendado para producción

***Licencias Creative Commons: Para obras creativas***

Las licencias Creative Commons (CC) complementan el ecosistema de software libre extendiéndolo a otros contenidos. Fundamentos:

- **CC BY (Atribución)**: La más permisiva, solo requiere crédito al autor
- **CC BY-SA (Atribución-CompartirIgual)**: Versión çopyleft"similar a GPL
- **CC BY-NC (NoComercial)**: Permite usos no comerciales
- **CC BY-ND (SinDerivadas)**: Permite redistribución sin cambios

**Importante**: Las licencias CC no son recomendadas para software, donde la GPL/LGPL son más apropiadas.

***Linus Torvalds: El creador revolucionario***

Linus Torvalds, nacido en 1969 en Helsinki, Finlandia, es una de las figuras más influyentes en la historia de la computación. Mientras estudiaba Ciencias de la Computación en la Universidad de Helsinki, Torvalds se vio frustrado por las limitaciones de los sistemas operativos disponibles, particularmente Minix, que aunque educativo, tenía restricciones en su uso y modificación. Esta frustración lo llevó en 1991, a la edad de 21 años, a embarcarse en el desarrollo de lo que sería el kernel Linux.

Su filosofía de desarrollo se caracteriza por dos principios fundamentales: Release early, release often"(publicar pronto y frecuentemente), que aceleró la innovación mediante la retroalimentación constante de la comunidad, y un modelo meritocrático donde las contribuciones técnicas valiosas tienen más peso que cualquier otra consideración. Aunque su estilo de liderazgo directo y sus comentarios francos han generado controversias, su enfoque ha demostrado ser extremadamente efectivo para mantener la coherencia técnica en proyectos masivos.

***El kernel Linux: Arquitectura y evolución***

El kernel Linux representa uno de los ejemplos más exitosos de software colaborativo en la historia. A diferencia del enfoque micronúcleo que adoptó GNU Hurd, Torvalds optó por una arquitectura monolítica modular, donde el núcleo principal maneja las funciones críticas del sistema, pero permite cargar módulos dinámicamente para expandir su funcionalidad. Esta decisión técnica, inicialmente criticada por algunos puristas, resultó ser clave para su adopción masiva.

Algunos hitos importantes en su desarrollo incluyen:

- Soporte inicial para arquitectura x86 (1991)
- Adopción del sistema de archivos ext (1992)
- Implementación del módulo de red (1994)
- Soporte para multiprocesamiento simétrico (1996)

Hoy, el kernel Linux soporta más de 30 arquitecturas de hardware diferentes y se actualiza con ciclos de lanzamiento cada 2-3 meses, con contribuciones de miles de desarrolladores en todo el mundo. Su impacto va desde servidores empresariales hasta sistemas embebidos en dispositivos IoT.

***Git: Revolucionando el control de versiones***

En 2005, Torvalds enfrentó otro desafío técnico significativo cuando la comunidad Linux necesitó reemplazar su sistema de control de versiones. Su solución fue Git, un sistema que diseñó en solo dos semanas de

trabajo intensivo, pero que revolucionaría la forma en que se gestiona el código fuente.

Git introdujo tres innovaciones fundamentales:

- Modelo distribuido donde cada copia es un repositorio completo
- Estructura de datos basada en grafos acíclicos dirigidos
- Mecanismos extremadamente eficientes para branching y merging

Estas características técnicas resolvieron problemas específicos que Torvalds había identificado en el desarrollo del kernel Linux, particularmente la necesidad de manejar miles de contribuciones paralelas de forma eficiente. Hoy, Git es el estándar indiscutido en control de versiones, con una adopción que supera el 90 % del mercado.

***El nacimiento de GNU/Linux***

El sistema operativo GNU/Linux surgió de la convergencia de dos proyectos históricos: el movimiento GNU iniciado por Richard Stallman en 1983 y el kernel Linux creado por Linus Torvalds en 1991. Stallman había visionado un sistema operativo completamente libre bajo la filosofía de software libre, desarrollando herramientas esenciales como el compilador GCC, el editor Emacs y el depurador GDB. Sin embargo, el proyecto GNU carecía de un kernel funcional cuando Hurd, su kernel original, demostró ser demasiado ambicioso en su diseño micronúcleo.

En 1991, Linus Torvalds, entonces un estudiante universitario en Finlandia, anunció su proyecto personal: un kernel monolítico inspirado en Minix pero libre de sus restricciones de licencia. Lo publicó bajo la licencia GPL de GNU, permitiendo su integración natural con las herramientas del proyecto GNU. Esta combinación fortuita -las herramientas de usuario de GNU con el kernel de Torvalds- formó el primer sistema operativo completamente libre y funcional.

La filosofía de desarrollo abierto de Torvalds resultó revolucionaria. A diferencia de los modelos cerrados predominantes, Linux creció mediante contribuciones globales coordinadas por Internet. Cada versión era mejorada por una comunidad internacional de desarrolladores, siguiendo el principio de release early, release often"(publicar pronto y frecuentemente). Para 1994, con la versión 1.0 del kernel, GNU/Linux ya demostraba ser viable para uso productivo, marcando el inicio de una nueva era en sistemas operativos.

***Distribuciones Linux: Diversidad y especialización***

Las distribuciones Linux (llamadas "distros") empaquetan el kernel Linux con software complementario para crear sistemas completos. Existen cientos de distros, cada una optimizada para diferentes necesidades:

- **Debian (1993)**: La distribución comunitaria por excelencia, conocida por su estabilidad y filosofía de software libre puro. Sirvió como base para muchas otras distros.
- **Red Hat Enterprise Linux (1994)**: Pionera en el modelo comercial de soporte empresarial, ampliamente usada en servidores corporativos.
- **Ubuntu (2004)**: Popularizó Linux en escritorios con su enfoque en facilidad de uso, derivada de Debian pero con ciclos de lanzamiento más frecuentes.
- **Arch Linux (2002)**: Filosofía "Keep It Simple"(KISS), con un sistema rolling-release y gran personalización.
- **Fedora (2003)**: Comunidad patrocinada por Red Hat que sirve como banco de pruebas para tecnologías innovadoras.
- **Slackware (1993)**: Una de las distribuciones más antiguas, conocida por su simplicidad y cercanía a Unix. No utiliza systemd y mantiene un enfoque altamente manual.
- **Gentoo (2002)**: Distribución para entusiastas que compila todo desde código fuente (Portage). Ofrece un control extremadamente granular sobre el sistema.
- **Void Linux (2008)**: Independiente, rolling-release, usa runit en lugar de systemd. Destaca por su ligereza y soporte nativo para musl libc.
- **Devuan (2014)**: Fork de Debian creado específicamente para evitar systemd, usando sysvinit o OpenRC en su lugar. Mantiene compatibilidad con paquetes Debian.
- **Artix Linux (2017)**: Derivada de Arch Linux pero sin systemd, ofrece múltiples opciones de init (OpenRC, runit, s6). Mantiene la filosofía rolling-release de Arch.

**Nota sobre distribuciones anti-systemd**: Slackware, Gentoo, Void, Devuan y Artix representan enfoques alternativos a systemd, ofreciendo sistemas con init tradicional (sysvinit), OpenRC, runit o s6. Estas distros son populares entre usuarios que prefieren simplicidad Unix tradicional o mayor control sobre el sistema de init.

**Nota sobre distribuciones anti-systemd**: Slackware, Gentoo, Void, Devuan y Artix representan enfoques alternativos a systemd, ofreciendo sistemas con init tradicional (sysvinit), OpenRC, runit o s6. Estas distros son populares entre usuarios que prefieren simplicidad Unix tradicional o mayor control sobre el sistema de init.

**Las distribuciones se diferencian principalmente por:**

- Sistema de gestión de paquetes (APT, RPM, Pacman)
- Ciclo de lanzamiento (fixed-release vs rolling-release)
- Selección de software predeterminado
- Filosofía de software (solo libre vs incluir propietario)
- Mercado objetivo (escritorio, servidor, etc)

El ecosistema de distribuciones demuestra la flexibilidad del modelo GNU/Linux, permitiendo adaptaciones para:

- Seguridad (Kali Linux para pentesting)
- Multimedia (Ubuntu Studio)
- Ciencia (Fedora Scientific)
- Dispositivos antiguos (Puppy Linux)
- IoT (Raspbian para Raspberry Pi)

Esta diversidad, unida al modelo de desarrollo colaborativo, ha hecho de GNU/Linux el sistema operativo más versátil del mundo, impulsando desde supercomputadoras hasta smartphones Android (que usan un kernel Linux modificado).

## Instalación de Sistemas Operativos

### *Preparación Inicial*

Antes de instalar cualquier sistema operativo, es fundamental:

1. **Verificar los requisitos del hardware**: - Consultar los requisitos mínimos del SO - Confirmar compatibilidad con UEFI/BIOS - Verificar espacio en disco disponible

2. **Preparar el medio de instalación**: - Descargar la imagen ISO oficial - Crear USB booteable usando herramientas como:

- Rufus (Windows)
- balenaEtcher (Multiplataforma)
- dd (Linux)

- ventoy (tener multiples .iso)

3. **Configurar el BIOS/UEFI**: - Acceder al menú de configuración (F2/DEL durante el arranque) - Establecer el orden de arranque - Habilitar/deshabilitar Secure Boot según necesidades

***Proceso General de Instalación***

El flujo típico de instalación sigue estos pasos:

1. **Iniciar desde el medio de instalación**: - Reiniciar el equipo - Seleccionar el dispositivo de instalación - Esperar que cargue el entorno de instalación

2. **Configuración básica**: - Seleccionar idioma y distribución del teclado - Establecer zona horaria - Crear usuario administrador

3. **Particionado del disco**:

- **Opción automática**: El instalador crea las particiones necesarias
- **Opción manual**: Permite control total sobre el esquema de particiones

4. **Selección de paquetes**: - Elegir entorno de escritorio (si aplica) - Seleccionar software adicional - Especificar si se incluirán controladores propietarios

5. **Instalación del gestor de arranque**: - Generalmente GRUB para sistemas Linux - Se instala típicamente en el MBR o partición EFI

6. **Finalización**: - El sistema solicitará reiniciar - Es importante retirar el medio de instalación - Al reiniciar, el sistema debería cargar normalmente

***Configuraciones Avanzadas***

Para usuarios con necesidades específicas:

1. **Cifrado del disco**: - Protege todos los datos con contraseña - Requiere configuración adicional durante el particionado

2. **LVM (Logical Volume Manager)**: - Permite gestión flexible del espacio en disco - Facilita redimensionamiento posterior

3. **RAID**: - Configuración de discos redundantes - Opciones RAID 0, 1, 5, etc. según necesidades

***Problemas Comunes y Soluciones***

- **El sistema no arranca después de instalar**: - Verificar orden de arranque en BIOS/UEFI - Reparar GRUB desde un live USB

- **No se detecta el disco duro**: - Verificar controladores necesarios - Cambiar modo SATA en BIOS (AHCI/IDE)
- **Pantalla negra al iniciar**: - Probar modos de video alternativos - Instalar controladores gráficos adecuados

***Consideraciones Finales***

- Siempre hacer backup de datos importantes antes de instalar
- Para sistemas dual-boot, instalar Windows primero
- Documentar las particiones creadas para referencia futura
- Considerar crear una partición separada para /home en Linux

## Virtualización

***Conceptos Fundamentales***

La virtualización permite ejecutar múltiples sistemas operativos (invitados) sobre un mismo hardware físico (anfitrión) mediante un *hypervisor*. Existen dos tipos principales:

- **Tipo 1 (Bare-metal)**: Ejecuta directamente sobre el hardware (ej: VMware ESXi, Microsoft Hyper-V)
- **Tipo 2 (Hosted)**: Se ejecuta como aplicación en el SO anfitrión (ej: VirtualBox, QEMU)

***VirtualBox: Virtualización para Todos***

Desarrollado por Oracle, es la solución más accesible para usuarios domésticos:

- Interfaz gráfica intuitiva (GUI)
- Soporte para múltiples SO invitados
- Características clave:
  - Snapshots (instantáneas del sistema)
  - Modo seamless (integrar ventanas del SO invitado)
  - Virtualización anidada (para contenedores dentro de VMs)

- Limitaciones:
  - Rendimiento inferior a soluciones basadas en KVM
  - Soporte limitado para GPU passthrough

***QEMU: El Emulador Versátil***

QEMU (Quick Emulator) es una herramienta de código abierto más potente:

- **Emulación completa**: Soporta múltiples arquitecturas (x86, ARM, RISC-V, etc.)
- **Aceleración con KVM**: Cuando se usa con módulos del kernel Linux (KVM), alcanza rendimiento casi nativo
- Modos de operación:
  - *System-mode*: Emula sistema completo (para VMs)
  - *User-mode*: Ejecuta binarios de otras arquitecturas

***KVM: Virtualización a Nivel de Kernel***

Kernel-based Virtual Machine transforma Linux en un hypervisor Tipo 1:

- Requisitos:
  - CPU con extensiones de virtualización (Intel VT-x/AMD-V)
  - Módulos del kernel: `kvm`, `kvm_intel` o `kvm_amd`
- Ventajas:
  - Rendimiento cercano al nativo
  - Soporte para muchas VMs simultáneas
  - Integración con libvirt para gestión
- Comandos básicos:

```
$ sudo virsh list --all
$ sudo virt-install --name=vm1 --ram=2048 --vcpus=2 \
  --disk path=/var/lib/libvirt/images/vm1.qcow2,size=20
```

***GPU Passthrough: Rendimiento Nativo***

Técnica para asignar hardware gráfico directamente a una VM:

- Requisitos complejos:
  - CPU y placa base compatibles con IOMMU (VT-d/AMD-Vi)
  - GPU dedicada para la VM
  - Configuración específica del kernel
- Proceso básico:
  1. Identificar grupos IOMMU: `lspci -nnk`
  2. Aislar la GPU en el kernel: `vfio-pci`
  3. Configurar XML de la VM (via `virsh edit`)
  4. Asignar dispositivos PCI a la VM
- Usos principales:
  - Gaming en máquinas virtuales
  - Estaciones de trabajo gráficas virtualizadas
  - Aplicaciones CAD/3D en VMs

***Comparativa de Tecnologías***

| **Software** | **Tipo** | **Rendimiento** | **Uso típico** |
|---|---|---|---|
| VirtualBox | Tipo 2 | Moderado | Usuario doméstico |
| QEMU (sin KVM) | Emulador | Bajo | Desarrollo multiplataforma |
| QEMU-KVM | Tipo 1 | Alto | Servidores/Estaciones |
| VMware ESXi | Tipo 1 | Muy alto | Entornos empresariales |

***Configuración Avanzada de QEMU***

Ejemplo de línea de comandos para máxima performance:

```
$ qemu-system-x86_64 \
 -enable-kvm -cpu host \
 -smp 8 -m 16G \
```

```
-drive file=vm.qcow2,format=qcow2 \
-device virtio-net,netdev=net0 \
-netdev user,id=net0 \
-vga none -nographic \
-device vfio-pci,host=01:00.0
```

Parámetros clave:

- `-enable-kvm`: Activa aceleración hardware
- `-cpu host`: Expone todas las características del CPU
- `-device vfio-pci`: Para GPU passthrough

***Consideraciones de Seguridad***

- Aislamiento de recursos críticos
- Configuración adecuada de redes virtuales
- Actualización regular del hypervisor
- Uso de TPM virtual para VMs (Windows 11)

## Ciberseguridad en la Era Digital

***La Importancia Estratégica de la Ciberseguridad***

En el contexto actual de hiperconectividad, la ciberseguridad ha evolucionado de ser un tema técnico a convertirse en un pilar estratégico para organizaciones y gobiernos. La digitalización masiva de procesos críticos y el aumento exponencial de dispositivos IoT han expandido la superficie de ataque, haciendo que las vulnerabilidades puedan tener consecuencias catastróficas. Solo en 2023, el costo global del cibercrimen superó los $8 trillones, superando el PIB de muchos países.

La ciberseguridad moderna debe abordar tres dimensiones clave:

- **Confidencialidad**: Protección contra accesos no autorizados
- **Integridad**: Garantía de que los datos no son alterados
- **Disponibilidad**: Asegurar el acceso continuo a sistemas

### *Modelo OSI: La Arquitectura Fundamental de Redes*

El modelo OSI (Open Systems Interconnection) desarrollado por ISO en 1984 sigue siendo el marco de referencia para entender las comunicaciones de red. Este modelo divide el proceso de comunicación en siete capas abstractas:

Tabla 1: Capas del Modelo OSI

| **Capa** | **Función** | **Ejemplo de Protocolos** |
|---|---|---|
| 7. Aplicación | Interfaz usuario | HTTP, FTP, DNS |
| 6. Presentación | Formato de datos | SSL/TLS, JPEG |
| 5. Sesión | Control de diálogos | NetBIOS, RPC |
| 4. Transporte | Confiabilidad | TCP, UDP |
| 3. Red | Enrutamiento | IP, ICMP |
| 2. Enlace | Acceso al medio | Ethernet, MAC |
| 1. Física | Transmisión binaria | Cableado, WiFi |

**. Análisis por Capas**

**Capa Física (1)**: Esta capa maneja la transmisión bruta de bits a través del medio físico. Incluye especificaciones de cables, conectores, señales eléctricas y wireless. Los ataques típicos incluyen cable tapping e interferencia electromagnética.

**Capa de Enlace (2)**: Responsable del direccionamiento físico (MAC) y control de acceso al medio. Tecnologías como Ethernet y WiFi operan aquí. Vulnerabilidades comunes son ARP spoofing y MAC flooding.

**Capa de Red (3)**: Gestiona el enrutamiento lógico mediante direcciones IP. Protocolos clave son IP, ICMP y BGP. Ataques frecuentes incluyen IP spoofing y DDoS.

**Capa de Transporte (4)**: Provee comunicación extremo-a-extremo con TCP (orientado a conexión) y UDP (sin conexión). Vulnerable a SYN floods y session hijacking.

### *Kali Linux: La Suite Definitiva para Pentesting*

Kali Linux representa el estándar de facto para pruebas de penetración ética. Desarrollado por Offensive Security, integra herramientas esenciales:

- `Metasploit Framework`: Para desarrollo y ejecución de exploits
- `Wireshark`: Analizador de protocolos de red
- `Burp Suite`: Testing de aplicaciones web
- `John the Ripper`: Password cracking

- `Nmap`: Escaneo de redes

Flujo de trabajo típico:

1. Reconocimiento con `nmap -sV -T4 192.168.1.0/24`
2. Escaneo de vulnerabilidades usando `nessus`
3. Explotación controlada con `msfconsole`
4. Post-explotación y análisis forense

***Principales Amenazas Cibernéticas Modernas***

Tabla 2: Panorama de Amenazas 2023

| **Amenaza** | **Crecimiento** | **Impacto** |
|---|---|---|
| Ransomware | +150 % | $20B |
| Phishing | +65 % | $4,9M/incidente |
| IoT Attacks | +112 % | 1.5M dispositivos/día |

**APT (Advanced Persistent Threats)**: Campañas sofisticadas como SolarWinds demostraron que incluso proveedores de software pueden convertirse en vectores de ataque, comprometiendo cadenas de suministro enteras.

**AI-Powered Attacks**: Uso de inteligencia artificial para crear phishing hiperpersonalizado o evadir sistemas de detección mediante adversarial machine learning.

***Estrategias Defensivas Contemporáneas***

La protección efectiva requiere un enfoque multicapa:

**Defensa Perimetral Avanzada**:

- Firewalls de última generación con inspección SSL
- Sistemas IDS/IPS con detección basada en comportamiento
- Segmentación de red microperimetral

**Arquitectura Zero Trust**:

- Verificación continua de identidad
- Principio de mínimo privilegio
- Microsegmentación de redes

**Automatización de Respuesta**:

- Plataformas SOAR (Security Orchestration, Automation and Response)
- Playbooks de respuesta automatizada
- Integración con SIEMs modernos

## Referencias

Andress, J. (2019). *Cybersecurity Essentials* (2nd). Jones & Bartlett Learning.

Lab, K. (2023). *What is Cybersecurity?* [Accessed: 2025-07-29]. https://www.kaspersky.com/resource-center/definitions/what-is-cyber-security

Stallings, W., & Brown, L. (2020). *Computer Security: Principles and Practice* (4th). Pearson.