🎓 Dispense Universitarie: Fondamenti di Sistemi Operativi e Architettura

Corso: Cyber Security & Ethical Hacking (S3L1) **Livello:** Master-Level / Tecnico Specialistico

---

1. Il Modello di Von Neumann

(Riferimento Slide 6-7)

1. 📖 Spiegazione Approfondita

Il modello di Von Neumann, proposto dal matematico John von Neumann negli anni '40, rappresenta l'architettura fondante della quasi totalità dei computer moderni. Prima di questo modello, i calcolatori erano spesso programmati meccanicamente o tramite cablaggi fissi. L'innovazione rivoluzionaria di Von Neumann è il concetto di **Stored-Program Computer**: sia le istruzioni del programma (software) che i dati su cui queste operano risiedono nella stessa memoria indirizzabile.

L'architettura si compone di tre sottosistemi principali interconnessi da un canale di comunicazione detto **BUS**:

1. **CPU (Central Processing Unit):** Il cervello del sistema, responsabile dell'elaborazione.
2. **Memoria Centrale:** Contiene istruzioni e dati (es. RAM).
3. **Periferiche di I/O (Input/Output):** Interfacce verso il mondo esterno (tastiera, monitor, dischi).

3. 🧠 Analisi Tecnica & Memorizzazione

- **Definizione Chiave:** Architettura in cui dati e istruzioni condividono lo stesso spazio di memoria e vengono trasferiti alla CPU tramite un Bus condiviso.
- **Collo di Bottiglia di Von Neumann:** Poiché il bus è condiviso, la velocità di elaborazione della CPU è limitata dalla velocità di trasferimento dei dati dalla memoria.
- **Contesto Reale:** Ogni volta che si analizza un buffer overflow, si sta sfruttando il fatto che dati e istruzioni risiedono nella stessa memoria (sovrascrivendo dati per alterare il flusso di istruzioni).

---

2. Hardware: Componenti di un Dispositivo

(Riferimento Slide 8-11)

1. 📖 Spiegazione Approfondita

Per compromettere o proteggere un sistema, è essenziale comprenderne la fisicità. I componenti principali includono:

- **Processore (CPU):** Esegue le istruzioni del software. I leader di mercato sono Intel (architettura x86) e AMD.
- **Scheda Madre (Motherboard):** Il circuito stampato principale che funge da "sistema nervoso", collegando CPU, RAM e periferiche tramite il Chipset.

- **Memoria RAM (Random Access Memory):** Memoria volatile e veloce utilizzata per i dati temporanei dei programmi in esecuzione. "Volatile" significa che perde i dati allo spegnimento.
- **Memoria di Archiviazione (Storage):**
  - *HDD (Hard Disk Drive):* Meccanico, magnetico, più lento ed economico.
  - *SSD (Solid State Drive):* Elettronico (memoria flash), molto più veloce, resistente agli urti, zero parti mobili.
- **Scheda Grafica (GPU):** Processore specializzato nel calcolo parallelo (rendering immagini). Può essere integrata (nella CPU) o dedicata (scheda a parte).
- **Alimentatore (PSU):** Converte la corrente alternata (AC) in continua (DC) per i componenti.
- **Unità Ottica:** Lettori CD/DVD/Blu-ray (in disuso, ma rilevanti per analisi forensi di vecchi supporti).
- **Sistema di Raffreddamento:** Ventole o liquido per dissipare il calore termico generato da CPU/GPU.
- **Interfacce:** Porte di espansione (USB, HDMI, Ethernet) per la connettività.

5. ⚠ Focus Esame

- **Differenza RAM vs Storage:** La RAM è veloce e volatile (lavoro attivo); lo Storage è lento e persistente (archiviazione).
- **Ruolo della GPU in Security:** Oltre alla grafica, le GPU sono usate massicciamente nel **Password Cracking** (es. Hashcat) per la loro capacità di calcolo parallelo.

---

3. Porte Logiche (Logic Gates)

(Riferimento Slide 12-14)

1. 📖 Spiegazione Approfondita

Le porte logiche sono i mattoni fondamentali dell'elettronica digitale. Operano su numeri binari (0 e 1, o Falso e Vero) secondo l'algebra di Boole. Ogni operazione logica complessa in una CPU è una combinazione di queste porte elementari.

Tavola delle Verità e Operatori:

- **NOT:** Inverte l'input ($0 \rightarrow 1$, $1 \rightarrow 0$).
- **AND:** Vero solo se **tutti** gli input sono veri (1 AND 1 = 1).
- **OR:** Vero se **almeno un** input è vero.
- **NAND (Not AND):** Opposto di AND (Vero se non sono tutti veri).
- **NOR (Not OR):** Opposto di OR (Vero solo se tutti sono falsi).
- **XOR (Exclusive OR):** Vero solo se gli input sono **diversi** (1 XOR 0 = 1). Fondamentale nella crittografia.
- **XNOR:** Opposto di XOR (Vero se gli input sono uguali).

3. 🧠 Analisi Tecnica & Memorizzazione

- **Contesto Reale:** L'operazione **XOR** è la base di molti cifrari simmetrici e stream cipher. In Assembly, fare `XOR EAX, EAX` è il metodo più veloce per azzerare un registro.

---

4. Il Sistema Operativo (OS) e il Kernel

(Riferimento Slide 15-19, 24-29)

1. 📖 Spiegazione Approfondita

Il Sistema Operativo è il software intermediario tra l'utente/applicazioni e l'hardware fisico. I suoi compiti cardine sono:

1. **Gestione Processi:** Scheduling e sincronizzazione dei programmi.
2. **Gestione Memoria:** Allocazione della RAM e Virtual Memory.
3. **File System:** Organizzazione logica dei dati su disco.
4. **I/O e Periferiche:** Comunicazione tramite driver.

Il Kernel: È il "nucleo" del sistema operativo. Risiede in memoria protetta ed è il solo componente che interagisce direttamente con l'hardware (Privilege Ring 0).

Architetture del Kernel:

- **Kernel Monolitico (es. Unix, Linux tradizionale):** Un unico file binario gigante che contiene gestione processi, memoria, driver e file system. *Pro:* Molto veloce. *Contro:* Se un driver crasha, l'intero sistema può andare in "Kernel Panic" (schermata blu/nera).
- **Microkernel (es. MINIX):** Kernel minimale. Driver e servizi girano nello spazio utente (User Space). *Pro:* Stabile (un crash del driver non blocca il sistema). *Contro:* Più lento a causa della continua comunicazione (message passing) tra User e Kernel space.
- **Kernel Ibrido (es. Windows NT/10/11, macOS):** Cerca un compromesso. Ha una struttura simile al microkernel ma esegue alcuni servizi critici (come lo stack grafico o di rete) nello spazio kernel per performance.
- **Modulare:** Estensione del monolitico (Linux moderno), dove moduli possono essere caricati/scaricati dinamicamente a runtime (es. inserendo una chiavetta USB si carica il modulo driver).

3. 🧠 Analisi Tecnica & Memorizzazione

- **Differenza Monolitico vs Microkernel:** Il primo è "tutto in uno" (performance), il secondo delega ai processi utente (stabilità).
- **Definizione Hybrid Kernel (Windows):** Combina la velocità del monolitico per sottosistemi critici con la modularità del microkernel per i driver non essenziali.

---

5. Avvio e Funzionamento (Focus Windows)

(Riferimento Slide 20-23)

1. 📖 Spiegazione Approfondita

Il ciclo di vita di un'applicazione segue tre fasi principali in un sistema Windows:

1. **Boot (Avvio):**
    - Il BIOS/UEFI inizializza l'hardware e carica il **Bootloader**.
    - Il Bootloader carica il **Kernel** in memoria.
    - Il Kernel inizializza i driver, le strutture dati e avvia il sottosistema grafico (GUI) e i servizi di sistema (es. `lsass.exe`, `services.exe`).
2. **Interazione Utente:**

- L'utente clicca un'icona. La GUI invia una richiesta al Kernel.
- Il Kernel crea un **Processo**: assegna un PID (Process ID), alloca memoria virtuale e risorse.

3. **Esecuzione:**
    - Il Kernel (tramite lo Scheduler) assegna tempo di CPU al processo.
    - Se l'app deve disegnare a schermo o salvare un file, esegue una **System Call** (chiamata di sistema) per chiedere al Kernel di farlo per lei.

4. 🔗 Cross-Connection Master

In ambito **Malware Analysis**, capire il processo di boot è vitale per analizzare la "persistenza" (come un virus si avvia automaticamente all'accensione, es. chiavi di registro Run o servizi).

---

6. Gestione della Memoria

(Riferimento Slide 30-41)

1. 📖 Spiegazione Approfondita

La memoria non è un blocco unico, ma una gerarchia piramidale basata su velocità e costo:

1. **Registri CPU:** Memoria interna al processore, istantanea ma minuscola.
2. **Cache (L1, L2, L3):** SRAM ultra-veloce che memorizza i dati più usati per evitare di interpellare la RAM.
3. **RAM (Memoria Centrale):** Lo spazio di lavoro principale.
4. **Memoria Secondaria (Disco):** Lenta, enorme, permanente.

Allocazione e Problemi: Quando più processi (Multi-tasking) chiedono memoria, l'allocazione sequenziale ("lineare") porta alla **Frammentazione**: si creano "buchi" di memoria libera troppo piccoli per essere usati da nuovi processi, sprecando RAM.

Soluzione: Paging (Paginazione) La memoria fisica è divisa in blocchi di dimensione fissa detti **Frame**. La memoria logica dei programmi è divisa in **Pagine (Pages)** della stessa dimensione.

- Il sistema operativo mappa le pagine logiche sui frame fisici in modo non contiguo.
- **Vantaggio:** Elimina la frammentazione esterna; il processo crede di avere memoria contigua, ma fisicamente è sparpagliata.

Memoria Virtuale (Swap) Se la RAM fisica finisce, il sistema operativo sposta le pagine meno usate ("Least Recently Used") su un file su disco (Swap file o Pagefile).

- *Effetto:* Permette di eseguire programmi più grandi della RAM fisica.
- *Costo:* Il disco è migliaia di volte più lento della RAM (Thrashing se si usa troppo swap).

5. ⚠ Focus Esame

- **Cos'è il Paging?** Tecnica per risolvere la frammentazione dividendo la memoria in blocchi di dimensione fissa.
- **Cos'è lo Swap?** Estensione della RAM su disco fisso.

---

7. Gestione dei Processi

(Riferimento Slide 42-51, 66)

1. 📖 Spiegazione Approfondita

Un **Processo** è un programma in esecuzione. Include: codice (text section), dati, stack, e stato dei registri.

Stati del Processo:

1. **Pronto (Ready):** In memoria, aspetta solo che la CPU si liberi.
2. **Esecuzione (Running):** La CPU sta eseguendo le sue istruzioni.
3. **Attesa (Waiting/Blocked):** Fermo in attesa di un evento esterno (es. input tastiera, lettura disco).
4. **Terminato:** Ha finito il lavoro, le risorse vengono liberate.

Scheduling e Multitasking:

- **Mono-tasking (es. MS-DOS):** Un solo processo alla volta. La CPU sta ferma (Idle) se il processo attende I/O. Inefficiente.
- **Multitasking Cooperativo:** Il processo deve cedere volontariamente il controllo. Se si blocca, blocca tutto il PC (Windows 3.1).
- **Multitasking Preemptive (Con Prelazione):** Standard moderno (Windows NT, Linux). Lo **Scheduler** del Kernel interrompe forzatamente il processo dopo un tempo prestabilito (**Time Slice** o **Quantum**) per dare spazio ad altri. Garantisce equità e reattività.
- **Time-Sharing:** Esecuzione ciclica rapida che dà all'utente l'illusione di parallelismo perfetto.

Sincronizzazione: Quando più processi/thread accedono a risorse condivise, servono meccanismi (semafori, mutex) per evitare conflitti o dati corrotti (**Race Conditions**) e blocchi infiniti (**Deadlock**).

---

8. Gestione Interfaccia Utente (UI) & Driver

(Riferimento Slide 52-54, 68)

1. 📖 Spiegazione Approfondita

L'OS gestisce l'input/output astraendo l'hardware.

- **Driver:** Software specifico fornito dal produttore che traduce le chiamate generiche dell'OS (es. "stampa pixel") in comandi elettrici specifici per quel dispositivo.
- **Buffer di I/O:** Aree di memoria temporanea che compensano la differenza di velocità tra CPU (velocissima) e periferiche (lente). Esempio: Buffer della tastiera che memorizza i tasti premuti se il sistema è momentaneamente occupato.
- **Interrupt:** Segnale hardware che avvisa la CPU di interrompere quello che sta facendo per gestire un evento (es. un tasto premuto o un pacchetto di rete arrivato).

---

9. File System

(Riferimento Slide 55-61)

1. 📖 Spiegazione Approfondita

Il File System (FS) è la struttura logica che permette di salvare, nominare e organizzare i dati persistentemente. Senza FS, il disco sarebbe solo una distesa di bit grezzi senza significato.

Funzioni:

- Mappatura dei file su blocchi fisici del disco.
- Gestione gerarchica (Cartelle/Directory).
- Gestione permessi (ACL) e metadati (data creazione, autore).

Tipologie Principali:

- **FAT32/ExFAT:** Semplice, universale (USB), niente permessi di sicurezza, limiti dimensione file (FAT32 max 4GB).
- **NTFS (New Technology File System):** Standard Windows. Supporta file enormi, compressione, crittografia (BitLocker), e ACL (sicurezza permessi).
- **EXT4 (Extended FS):** Standard Linux. Robusto, performante (Journaling).
- **HFS+/APFS:** Standard Apple/macOS.
- **NFS/SMB:** File system di rete per accedere a file remoti.

Struttura Generica:

- **Superblock:** Contiene i metadati globali del FS (dimensione totale, blocchi liberi).
- **Inode (Linux) / MFT Record (Windows):** La "carta d'identità" del file. Contiene permessi, date e puntatori ai blocchi di dati, ma *non* il nome del file (che sta nella directory).

---

10. Architettura Windows e PowerShell

(Riferimento Slide 62-74, 82-96)

1. 📖 Spiegazione Approfondita

Architettura User vs Kernel Mode: Per sicurezza, Windows divide la memoria in due:

- **Kernel Mode (Ring 0):** Accesso totale all'hardware. Qui girano Kernel, HAL e Driver. Un crash qui ferma il sistema (BSOD).
- **User Mode (Ring 3):** Accesso limitato. Qui girano applicazioni (Word, Chrome) e servizi. Non possono toccare l'hardware direttamente.
- **Sottosistemi DLL (es. `kernel32.dll`, `ntdll.dll`):** Fungono da interfaccia. Quando un'app vuole creare un file, chiama una funzione DLL pubblica, che valida la richiesta ed esegue il "Context Switch" verso il Kernel Mode per eseguire l'azione reale.

PowerShell (PS): È la shell avanzata di Microsoft. A differenza di Bash (che manipola testo), PowerShell è **Object-Oriented**: i comandi restituiscono oggetti .NET con proprietà e metodi.

2. 🛠 Sintassi & Comandi (Cleaning)

Struttura Cmdlet: `Verbo-Sostantivo` (es. `Get-Service`).

Comando Slide (Corretto)	Alias	Descrizione Tecnica
`Get-Command`	-	Elenca tutti i comandi disponibili.
`Get-ChildItem`	`ls`, `dir`	Elenca file e cartelle nella directory corrente.
`Set-Location [path]`	`cd`	Cambia la directory di lavoro.
`Copy-Item [src] [dst]`	`cp`, `copy`	Copia file o oggetti.
`Remove-Item [file]`	`rm`, `del`	Cancella file o oggetti.
`Get-Content [file]`	`cat`, `type`	Legge il contenuto di un file (utile per leggere log).
`Get-Process`	`ps`	Mostra i processi in esecuzione.
`Select-Object`	-	Filtra le proprietà di un oggetto (fondamentale per pulire l'output).
`Where-Object`	`?`	Filtra gli oggetti in base a condizioni logiche.

Pipe (`|`): Passa l'oggetto output di un comando come input al successivo. *Esempio:* `Get-Service | Where-Object {$_.Status -eq 'Stopped'}` (Trova tutti i servizi fermi).

5. ⚠ Focus Esame

- **Perché esistono le DLL di sistema?** Per impedire alle applicazioni di accedere direttamente al Kernel (sicurezza e stabilità).
- **Differenza Bash vs PowerShell:** Bash gestisce stringhe di testo; PowerShell gestisce Oggetti .NET.

---

11. Linux: Struttura e Fondamenti

(Riferimento Slide 75-86)

1. 📖 Spiegazione Approfondita

Nato nel 1991 da Linus Torvalds (ispirato a MINIX/Unix), Linux è un sistema operativo con kernel monolitico, open source e gratuito. È dominante nei server, cloud e dispositivi embedded (Android).

Gerarchia del File System (FHS): In Linux tutto parte dalla radice `/` (Root). Non esistono lettere di unità (C:, D:) come in Windows.

- `/bin` & `/sbin`: Binari (eseguibili) essenziali (comandi utente vs comandi admin).
- `/etc`: File di configurazione del sistema (es. `/etc/passwd`).
- `/home`: Cartelle personali degli utenti (simile a `C:\Users`).
- `/var`: Dati variabili, specialmente **LOG** (`/var/log`).
- `/dev`: Device files. In Linux "tutto è un file", anche l'hard disk (`/dev/sda`) o il terminale (`/dev/tty`).
- `/tmp`: File temporanei (svuotata al riavvio).
- `/root`: La home directory dell'utente amministratore (root).

Gestione Utenti e Sudo:

- **Root (UID 0):** Il superutente con poteri assoluti.
- **Utenti Standard (UID >= 1000):** Poteri limitati.
- `/etc/passwd`**:** Database leggibile da tutti contenente gli utenti, i loro UID, GID e la shell di default.
- `/etc/shadow`**:** Contiene le password (hashate). Leggibile solo da root.

Comando `sudo` (SuperUser DO): Permette a un utente autorizzato di eseguire un comando con privilegi di root senza dover fare login come root. È più sicuro perché traccia l'azione nei log e richiede la password dell'utente, non quella di root.

3. 🧠 Analisi Tecnica & Memorizzazione

- **Creazione Processi (Fork):** In Linux, un processo si crea clonando se stesso tramite la chiamata `fork()`. Il processo originale è il **Padre**, il nuovo è il **Figlio**. Hanno PID diversi.
- **Differenza Processo vs Thread:**
  - *Processo:* Memoria isolata. Pesante da creare.
  - *Thread:* "Processo leggero". Condivide la memoria con gli altri thread dello stesso processo. Se un thread corrompe la memoria, danneggia l'intero processo.

5. ⚠ Focus Esame

- **A cosa serve `/dev`?** Contiene i file che rappresentano le periferiche hardware.
- **Qual è l'UID di Root?** 0.
- **Cos'è la `fork()`?** La system call per creare un nuovo processo duplicando quello esistente.