

Aspetti Legali e Deontologici (Riferimenti Normativi)

1. 📖 Spiegazione Approfondita

Prima di intraprendere qualsiasi attività tecnica, è imperativo comprendere il quadro normativo italiano. L'attività di *Ethical Hacking* si distingue dal crimine informatico esclusivamente per la presenza di **autorizzazione**. Senza il consenso esplicito del proprietario del sistema, qualsiasi test di sicurezza è illegale.

Il Codice Penale Italiano punisce severamente le violazioni tramite due articoli cardine:

- **Articolo 615-ter (Accesso abusivo a un sistema informatico o telematico):** Punisce chiunque si introduca in un sistema protetto o vi rimanga contro la volontà del titolare. La pena base è la reclusione fino a 3 anni, ma aumenta (1-5 anni) se il fatto è commesso da un operatore di sistema o pubblico ufficiale (abuso di poteri).
- **Articolo 635-bis (Danneggiamento di informazioni, dati e programmi informatici):** Punisce chi distrugge, deteriora, cancella o altera dati altrui. La pena base va da 6 mesi a 3 anni.

In ambito scolastico ed etico, vige una **Clausola di Non Responsabilità**: le tecniche apprese devono essere utilizzate solo in ambienti controllati (laboratori) o autorizzati. Ogni studente è responsabile delle proprie azioni legali.

3. 🧠 Analisi Tecnica & Memorizzazione

- **Definizione Chiave:** L'Ethical Hacking richiede sempre un mandato scritto. La differenza tra un Pentester e un criminale non è negli strumenti usati, ma nel **permesso**.
- **Contesto Reale:** In fase di assunzione o contratto B2B, la prima cosa che viene richiesta/firmata è una NDA (Non-Disclosure Agreement) e una lettera di ingaggio che manleva il tester da azioni che simulano attacchi, purché restino nel perimetro definito (Scope).

5. ⚠️ Focus Esame

- **Domanda frequente:** Qual è la differenza tra l'art. 615-ter e il 635-bis?
 - *Risposta:* Il 615-ter riguarda l'**accesso** (entrare/rimanere), il 635-bis riguarda il **danneggiamento** (distruggere/alterare dati).

Introduzione al Penetration Testing

1. 📖 Spiegazione Approfondita

Il **Penetration Testing (Pentesting)** è un processo metodico volto a valutare la sicurezza di un sistema simulando attacchi reali. Non si tratta di hacking casuale, ma di una procedura strutturata per evidenziare carenze nei sistemi informativi.

Lo studio di questa disciplina serve a:

1. **Sviluppare Competenze Pratiche:** Passare dalla teoria all'azione tecnica.
2. **Comprendere le Vulnerabilità:** Capire come nascono i bug per poterli prevenire (difesa proattiva).
3. **Contribuire alla Sicurezza:** Proteggere infrastrutture e dati della comunità.

Al termine del percorso, l'obiettivo è padroneggiare concetti come Scanner, Vulnerability Scanner, Enumeration, Exploitation e l'uso avanzato di Kali Linux.

3. 🧠 Analisi Tecnica & Memorizzazione

- **Definizione Chiave:** Un Penetration Test è una simulazione autorizzata di un attacco informatico utilizzata per valutare la sicurezza del sistema.

Le Fasi del Penetration Testing (Ciclo di Vita)

1. 📖 Spiegazione Approfondita

Un Penetration Test professionale non è un evento singolo, ma un ciclo composto da fasi distinte e sequenziali. Saltare una fase compromette la qualità del risultato finale.

Le fasi sono:

1. **Fase 0 - Ingaggio (Engagement):** Definizione degli accordi legali e operativi.
2. **Fase 1 - Ricerca delle Informazioni (Information Gathering):** Raccolta dati sull'obiettivo (OSINT).
3. **Fase 2 - Enumerazione e Scansione:** Identificazione tecnica di porte, servizi e vulnerabilità.
4. **Fase 3 - Exploit:** Sfruttamento delle vulnerabilità per ottenere l'accesso.
5. **Fase 4 - Mantenimento Accesso & Scalata Privilegi:** Persistenza e ottenimento di diritti amministrativi.
6. **Fase 5 - Analisi e Report:** Documentazione dei risultati e raccomandazioni di sicurezza.

5. ⚠️ Focus Esame

- **Trabocchetto:** Spesso si confonde la fase di "Scansione" con quella di "Information Gathering". La *Gathering* può essere passiva (non tocca il server), la *Scansione* è attiva (invio pacchetti al server).

Fase 0: Fase di Ingaggio (Pre-Engagement)

1. 📖 Spiegazione Approfondita

Questa è la fase preliminare, prettamente burocratica e strategica, dove si definiscono le "regole del gioco". Elementi fondamentali definiti in questa fase:

- **Costi:** Budget allocato per l'attività.
- **Perimetro (Scope):** Quali asset (IP, domini, applicazioni) sono oggetto del test e quali sono *tassativamente esclusi*.
- **Finestra Temporale:** Quando avverrà il test (es. "dal 1 al 15 del mese", "dalle 18:00 alle 22:00 per non impattare la produzione").
- **Aspetti Legali:** Gestione dati sensibili (GDPR) e manleva.
- **Regole di Ingaggio (RoE):** Cosa è permesso fare (es. DoS è vietato? Social Engineering è permesso?).
- **Backup:** Verifica essenziale dei backup prima di lanciare exploit potenzialmente distruttivi.
- **Metodologia:** Scelta tra Black/White/Grey Box.

Si distinguono inoltre due macro-categorie di test:

- **Web Application PT:** Focus su vulnerabilità applicative (SQLi, XSS) tramite browser/proxy.
- **PT Infrastrutturale:** Focus su server, router, firewall, servizi di rete e OS.

3. 🧠 Analisi Tecnica & Memorizzazione

- **Contesto Reale:** Un errore nello *Scope* può portare a testare il server sbagliato (es. quello di un altro cliente in hosting condiviso), con conseguenze legali disastrose.

Metodologie di Test: Black, White e Grey Box

1. 📖 Spiegazione Approfondita

Esistono tre approcci principali basati sulla quantità di informazioni fornite al tester prima dell'inizio delle attività:

1. Black Box Testing (Blind Testing):

- **Info:** Nessuna conoscenza preliminare. Si ha solo il nome dell'azienda o l'URL.
- **Obiettivo:** Simulare un attacco realistico da parte di un hacker esterno.
- **Pro:** Realismo massimo.
- **Contro:** Richiede molto tempo (reconnaissance), rischia di non coprire vulnerabilità interne profonde.

2. White Box Testing (Crystal Box):

- **Info:** Accesso completo (codice sorgente, schemi di rete, credenziali, regole firewall).
- **Obiettivo:** Audit completo e approfondito (spesso usato per Code Review).
- **Pro:** Copertura totale delle vulnerabilità, ottimizzazione dei tempi.
- **Contro:** Meno realistico (non simula lo sforzo di intrusione esterna).

3. Grey Box Testing:

- **Info:** Accesso parziale (es. credenziali utente standard, diagrammi logici parziali).
- **Obiettivo:** Simulare un attacco da parte di un "Insider" (dipendente infedele) o di un hacker che ha violato il perimetro esterno.
- **Pro:** Ottimo bilanciamento tra realismo e profondità di analisi.

5. ⚠️ Focus Esame

- **Domanda:** Quale metodologia useresti per simulare un dipendente che vuole rubare dati?
 - *Risposta:* Grey Box (Insider threat).

Fase 1: Raccolta Informazioni (Information Gathering)

1. 📖 Spiegazione Approfondita

Primo passo operativo. L'obiettivo è mappare la superficie di attacco. Più informazioni si raccolgono, più facile sarà l'exploit. Si cercano dati su: Investitori, Impiegati (per phishing), Sedi fisiche, Dirigenti, Infrastruttura di Rete.

Tecnicamente, si mira a definire per ogni IP:

- È attivo? (Host Discovery)
- È client o server?
- Che OS usa? (Linux, Windows)
- Per il web: Sottodomini, tecnologie usate (PHP, Java, CMS).

Esistono due modalità di raccolta:

1. **Passiva:** Nessun contatto diretto con il bersaglio. Si usano fonti pubbliche (Google, Whois, Social Network). Il target non sa di essere osservato.
2. **Attiva:** Interazione diretta con i sistemi (es. Interrogazione DNS, Ping, Traceroute). Il target potrebbe rilevare l'attività nei log.

3. 🧠 Analisi Tecnica & Memorizzazione

- **Definizione Chiave (OSINT):** Open Source Intelligence. L'arte di raccogliere dati da fonti pubblicamente accessibili.
- **Contesto Reale:** Spesso si ottengono credenziali valide dai leak pubblici o dai post LinkedIn dei dipendenti, rendendo inutili exploit complessi.

Fase 2: Enumerazione e Scansione

1. 📖 Spiegazione Approfondita

In questa fase si passa dall'identificare *chi* è il target a capire *cosa* offre il target. L'**Enumerazione** serve a evitare di lanciare exploit a caso (es. exploit Windows contro un server Linux). La **Scansione** determina:

1. Host attivi.
2. Porte aperte (punti di ingresso).
3. Servizi in esecuzione (Apache, SSH, SMB).
4. Versioni software e Sistemi Operativi.

Tipi di Scansione (con riferimento a **nmap**):

- **TCP Connect Scan:** Completa il Three-Way Handshake. Affidabile ma rumoroso (lascia molti log).
- **SYN Scan (Stealth):** Invia SYN, riceve SYN-ACK, ma risponde con RST. Non completa la connessione. Più veloce e meno visibile.
- **UDP Scan:** Scansiona servizi UDP (DNS, SNMP). Lento e complesso perché l'UDP è connectionless.

Stati delle porte:

- **Aperta:** Il servizio risponde (accetta connessioni).
- **Chiusa:** Il sistema risponde "non c'è nessuno qui" (RST).
- **Filtrata:** Nessuna risposta. Probabilmente un Firewall sta bloccando il pacchetto (drop).

2. ✂ Sintassi & Comandi (Cleaning)

Anche se non esplicitati nelle slide in dettaglio codice, i concetti si riferiscono a:

- `nmap -sT [target]` (TCP Connect)
- `nmap -sS [target]` (SYN Scan - richiede privilegi root)
- `nmap -sU [target]` (UDP Scan)

Fase 2b: Valutazione delle Vulnerabilità (Vulnerability Assessment)

1. 📖 Spiegazione Approfondita

Il **Vulnerability Assessment (VA)** mira a creare una lista (inventario) di vulnerabilità note presenti sui sistemi. Si utilizzano **Vulnerability Scanner** (strumenti automatici come **Nessus** o OpenVAS).

- **Funzionamento:** Lo scanner interroga il target, confronta le risposte (banner, versioni) con un database di vulnerabilità note (CVE) e genera un report.
- **Configurazione:** È cruciale configurare lo scanner per evitare falsi positivi o di mandare in crash il servizio target per troppe richieste.
- **Analisi Manuale:** Necessaria dopo l'automazione per verificare i risultati (triage dei falsi positivi).

3. 🧠 Analisi Tecnica & Memorizzazione

- **Differenza VA vs PT:** Il VA elenca le porte aperte e le possibili falle ("Potrebbe essere vulnerabile"). Il PT sfrutta quelle falle ("Sono entrato").
- **Nessus:** Standard industriale per il VA.

Differenze Chiave: Scanner vs VA vs Pentester

1. 📖 Spiegazione Approfondita

È fondamentale distinguere i ruoli e gli strumenti:

1. **Network Scanner (es. Nmap):** Mappa la topologia. Dice *cosa c'è* (IP, Porte, Servizi). Non cerca bug di sicurezza, cerca asset.
2. **Vulnerability Scanner (es. Nessus):** Identifica *potenziali debolezze* (patch mancanti, misconfigurazioni). Genera report automatici.
3. **Vulnerability Assessment (Processo):** È l'intero processo di identificazione, classificazione e prioritarizzazione dei rischi (include scansione + analisi manuale).
4. **Penetration Tester (Persona/Ruolo):** Usa tutti gli strumenti sopra, ma va oltre. Esegue **Exploitation** (sfruttamento), scala i privilegi e dimostra l'impatto reale di un attacco (Business Impact).

5. ⚠️ Focus Esame

- Se ti viene chiesto di "trovare e sfruttare" una falla, sei un Pentester. Se devi solo "elencare le patch mancanti", stai facendo un Vulnerability Assessment.

Fase 3: Exploit

1. 📖 Spiegazione Approfondita

È la fase attiva dell'attacco. Si verifica se le vulnerabilità ipotizzate sono reali. Si divide in:

- **Exploit OS:** Attacchi a livello Kernel o servizi di sistema.
- **Exploit Web:** Attacchi alle applicazioni (rif. OWASP Top 10, es. SQLi, XSS).
- **Exploit Network:** Attacchi ai protocolli di rete (es. MITM).

Concetto di POC (Proof of Concept): Una POC è la documentazione tecnica che dimostra che una vulnerabilità è sfruttabile. Una POC valida deve essere:

1. **Chiara:** Comprensibile.
2. **Dettagliata:** Riproducibile (chiunque segua i passi ottiene lo stesso risultato).
3. **Pratica:** Deve dimostrare l'impatto (es. "ho letto il file /etc/passwd").

4. 🔗 Cross-Connection Master

Integrazione con **Metasploit Framework:** È il tool principale in Kali Linux per gestire ed eseguire exploit in questa fase.

Fase 4: Mantenimento Accesso, Scalata Privilegi e Backdoor

1. 📖 Spiegazione Approfondita

Una volta entrati ("pwned"), l'obiettivo è restare e diventare amministratori.

1. Mantenimento (Persistence): Garantirsi l'accesso futuro anche se il sistema viene riavviato o la falla originale viene patchata.

- Metodi: Creazione utenti nascosti, modifica regole firewall.
- **Backdoor:** Punto di accesso segreto che bypassa l'autenticazione normale.
 - *Software:* Trojan, Rootkit (nascondono la loro presenza al sistema operativo).
 - *Hardware:* Chip modificati (raro, supply chain attack).
 - *Integrate:* Credenziali hardcoded lasciate dagli sviluppatori.

2. Scalata dei Privilegi (Privilege Escalation):

- **Verticale:** Da utente standard (**user**) a superutente (**root/Administrator**). Sfrutta bug del kernel o misconfigurazioni.
- **Orizzontale:** Da utente A a utente B (stesso livello, ma dati diversi). Spesso tramite furto credenziali o session hijacking.

5. ⚠️ Focus Esame

- **Domanda:** Qual è la differenza tra un Trojan e un Rootkit?
 - *Risposta:* Il Trojan è un software malevolo che si finge legittimo. Il Rootkit è progettato specificamente per *nascondere* la presenza di malware e processi agli occhi del sistema operativo e dell'antivirus.

Fase 5: Reporting

1. 📖 Spiegazione Approfondita

Il prodotto finale del lavoro non è l'hack, ma il **Report**. Senza un buon report, il test è inutile per il cliente. Deve contenere:

1. Tecniche utilizzate.
2. Vulnerabilità trovate (con classificazione di rischio).
3. Vettori di attacco (come sono state sfruttate).
4. **Remediation Action**: Come correggere (patching, cambio configurazione).

Kali Linux: Introduzione e Gestione Pacchetti

1. 📖 Spiegazione Approfondita

Kali Linux è la distribuzione standard per il pentesting, basata su **Debian**. Contiene centinaia di tool preinstallati.

- **Credenziali di default**: Username: **kali**, Password: **kali**.

Gestione Pacchetti (Package Management): In Linux non si scaricano **.exe**. Si usano repository centralizzati.

- **dpkg**: Gestore base per pacchetti **.deb**. Non gestisce le dipendenze (le librerie necessarie al software).
- **APT (Advanced Packaging Tool)**: Wrapper avanzato su dpkg. Scarica i pacchetti dai repository definiti in **/etc/apt/sources.list** e risolve automaticamente le dipendenze.

2. ✂ Sintassi & Comandi (Cleaning)

- **Installazione **.deb** locale**:

```
dpkg -i nome_pacchetto.deb
```

- **Rimozione **.deb****:

```
dpkg -r nome_pacchetto
```

- **Installazione da Repo (Consigliato)**:

```
apt install nome_pacchetto
```

- **Rimozione da Repo**:

```
apt remove nome_pacchetto
```

Shell Linux: Navigazione e File System

1. 📖 Spiegazione Approfondita

Saper usare la CLI (Command Line Interface) è obbligatorio. I permessi file (es. **drwxr-xr-x**) indicano: Tipo (d=directory), Permessi Proprietario, Permessi Gruppo, Permessi Altri. (r=read, w=write, x=execute).

2. ✂ Sintassi & Comandi (Cleaning)

Ecco i comandi corretti (correzione errori OCR dalle slide):

- **pwd**: Print Working Directory (Dove sono?).
- **ls**: List (Elenca file).
 - **ls -l**: Formato lungo (permessi, proprietario, dimensione).
 - **ls -la**: Mostra anche file nascosti (che iniziano con **.**).
- **cd [path]**: Change Directory.
 - **cd ..**: Sale di un livello.
 - **cd /**: Va alla root.
 - **cd ~** (o solo **cd**): Va alla home dell'utente.
- **touch [file]**: Crea un file vuoto.
- **cp [sorgente] [destinazione]**: Copia file.
 - **cp -r**: Copia ricorsiva (per cartelle).
- **mv [sorgente] [destinazione]**: Sposta o Rinomina file.
- **mkdir [nome]**: Crea cartella.
- **rm [file]**: Rimuove file (Definitivo, niente cestino!).
 - **rm -r [cartella]**: Rimuove cartella e contenuto.

5. ⚠ Focus Esame

- **Attenzione:** **rm** in terminale cancella definitivamente. Non esiste "annulla" o cestino di default nella shell.

Shell Linux: Networking

1. 📖 Spiegazione Approfondita

Comandi per analizzare la configurazione di rete e le connessioni.

2. ⚡ Sintassi & Comandi (Cleaning)

- **ifconfig** (obsoleto ma usato) o **ip a** (moderno): Mostra interfacce (**eth0**, **lo**), indirizzi IP (**inet**), Netmask e MAC Address (**ether**).
- **netstat** o **ss**: Mostra connessioni di rete, porte in ascolto e tabelle di routing.
 - Spesso l'output è lungo, quindi si usa la **PIPE (|)** per filtrare.
 - Esempio: **netstat -antp | grep apache** (Cerca solo connessioni legate ad Apache).

3. 🧠 Analisi Tecnica & Memorizzazione

- **Loopback (lo)**: Interfaccia virtuale (127.0.0.1). Serve al computer per "parlare con se stesso".
- **Pipe (|)**: Prende l'output del comando a sinistra e lo usa come input per il comando a destra.

Gestione Processi e Servizi

1. 📖 Spiegazione Approfondita

In Linux, ogni programma in esecuzione è un processo identificato da un **PID** (Process ID).

2. 🗄️ Sintassi & Comandi (Cleaning)

- **Servizi:**
 - `service [nome_servizio] start` (Avvia)
 - `service [nome_servizio] stop` (Ferma)
 - `service --status-all` (Lista stato servizi: + attivo, - fermo).
 - **Processi:**
 - `ps`: Processi della shell corrente.
 - `ps aux`: Tutti i processi di tutti gli utenti.
 - `top`: Task manager dinamico (aggiorna real-time CPU/RAM).
 - `kill [PID]`: Termina forzatamente un processo (non presente nelle slide ma fondamentale contesto `ps`).
-

Strumenti dell'Ethical Hacker (Tool Overview)

1. 📖 Spiegazione Approfondita & Comandi

1. Netcat (`nc`)

Definito il "Coltellino Svizzero" (Swiss Army Knife). Legge e scrive dati su connessioni di rete TCP/UDP.

- **Client Mode (Connessione):** `nc [IP] [Porta]`
 - Utile per: Banner Grabbing (leggere versione servizio), test porta aperta.
- **Server Mode (Ascolto):** `nc -l -p [Porta]`
 - Utile per: Ricevere connessioni (es. Reverse Shell), chat semplice, trasferimento file.

2. Tcpdump

Sniffer di pacchetti da riga di comando (CLI). Cattura il traffico grezzo.

- Utile per: Debug rapido su server remoti senza interfaccia grafica.

3. Wireshark

Analizzatore di protocollo grafico (GUI). Lo standard de facto per l'analisi di rete.

- **Funzioni:** Cattura pacchetti, filtraggio avanzato, ricostruzione flussi TCP (Follow TCP Stream).
- **Utilizzo:** Selezionare interfaccia (`eth0`), avviare cattura, analizzare dettagli (Source, Destination, Protocol, Info).

4. Burp Suite

Piattaforma integrata per il test di sicurezza delle **Web Applications**.

- **Architettura:** Agisce come un **Proxy** tra il browser e il server web. Intercetta e modifica le richieste HTTP "al volo".
- **Moduli principali:**
 - **Proxy:** Intercetta traffico (Forward/Drop).
 - **Scanner:** Vulnerability scanner automatico (solo versione Pro).

- **Intruder:** Fuzzing e Brute-force (automazione attacchi custom).
- **Repeater:** Ripete singole richieste modificandole manualmente (fondamentale per analisi granulare).
- **Sequencer:** Analizza la casualità dei token di sessione.
- **Workflow:** Configurare proxy browser -> Intercettare richiesta -> Inviare a Repeater -> Modificare parametri -> Analizzare Risposta.