ESERCITAZIONE FINALE
(21 gennaio 2026)

Corso: **3770/10840604-011/606/DEC/25**
Titolo: **ESPERTO IN SICUREZZA INFORMATICA – ED. ROVIGO**
Sede: **ROVIGO (RO), Via N. Badaloni 2**

Modulo 3: **MONITORAGGIO DELLA SICUREZZA DEL SISTEMA INFORMATIVO**

Docente: Davide Gessi

Corsista: Francesco Baldo

Valutazione: 10/10

Securing Software Esercitazione

1. **Attraverso quali tecniche un attore malevolo potrebbe “craccare” un software, cioè bypassare la registrazione o il pagamento per poterlo usare gratuitamente?**
2. **Distingui la natura dei due tipi di attacchi “cross-site” discussi:**
   - Cross-Site Scripting (XSS)
   - Cross-Site Request Forgery (CSRF)
3. **Perché è necessario “fare l'escape” (cioè neutralizzare) alcuni caratteri nei dati di input?**
4. **Nel contesto di SQL, che cos’è una prepared statement (istruzione preparata)?**
5. **Perché la validazione lato client (client-side validation) è considerata meno sicura rispetto a quella lato server (server-side validation)?**
6. **Riferimento alla vignetta GeekHero**
   Dal punto di vista pratico, la vignetta sopra ha probabilmente ragione nel rappresentare il comportamento della maggior parte degli utenti verso il software open-source.
   Tuttavia, anche se questo fosse la tua opinione, perché potrebbe comunque essere una buona idea usare (o sviluppare) più software open-source, dal punto di vista della sicurezza informatica?
7. **In che modo i package manager (come apt, yum, npm, ecc.) sono simili agli app store (Apple App Store, Google Play Store, Microsoft Store, ecc.) dal punto di vista della cybersecurity?**
8. **Contro quale tipo di minaccia aiuta a difendersi l’uso del campo Content-Security-Policy (CSP) nel nostro codice sorgente?**
9. **Fornisci un esempio concreto di una situazione in cui potresti voler usare il metodo HTTP POST invece del metodo GET.**
10. **Heartbleed (CVE-2014-0160)**
    Il bug noto come Heartbleed, scoperto nel 2014, generò un’enorme preoccupazione su Internet: fu uno dei primi casi in cui una vulnerabilità informatica venne diffusa anche dai media generalisti, mentre i ricercatori di sicurezza cercavano di avvisare il pubblico e incoraggiare un aggiornamento urgente dei sistemi.
    Leggi informazioni su Heartbleed, ad esempio dalla pagina Wikipedia o da altre fonti affidabili (come un video divulgativo).
    Perché Heartbleed rappresentava una minaccia così grave per la sicurezza degli utenti?
    Qua la vignetta obbligatoria xkcd

Risposte:

1 Un attore malevolo può craccare un software usando tecniche come il reverse engineering (analizzare il codice per capire come funziona il controllo della licenza), la modifica del binario (patch per saltare i controlli), l'uso di keygen o seriali falsi, oppure l'intercettazione e la manipolazione delle comunicazioni con il server di attivazione. **Ottimo.** Completo e centrato, anche l'attivazione server

2 XSS: è un attacco in cui l'attaccante riesce a iniettare codice JavaScript malevolo in una pagina web, che viene poi eseguito nel browser di altri utenti.
CSRF: è un attacco in cui l'attaccante induce un utente autenticato a eseguire, a sua insaputa, una richiesta verso un sito web (ad esempio cambiare una password), sfruttando la fiducia del sito nel browser dell'utente. **Corretto.** Definizioni pulite di XSS e CSRF, nessuna confusione

3 Fare l'escape dei caratteri serve a evitare che l'input dell'utente venga interpretato come codice. In questo modo si prevengono attacchi come SQL injection o XSS, dove caratteri speciali potrebbero cambiare il significato del comando eseguito. **Giusto.** Forse un po' generico, ma il concetto chiave c'è.

4 Una prepared statement è un'istruzione SQL in cui la struttura della query è definita in anticipo e i dati vengono passati separatamente. Questo impedisce che l'input dell'utente venga interpretato come parte del codice SQL, aumentando la sicurezza. **Ottimo.** Spiegazione chiara del perché le prepared statement funzionano.

5 La validazione lato client è meno sicura perché può essere facilmente bypassata (ad esempio disattivando JavaScript o modificando le richieste). La validazione lato server è più affidabile perché avviene in un ambiente controllato dal server e non dall'utente, quindi più difficilmente penetrabile. **Perfetto**

6 Anche se molti utenti non leggono il codice open-source, usarlo può essere una buona idea per la sicurezza perché il codice è pubblico e può essere controllato da molti sviluppatori e ricercatori. Questo aumenta la probabilità che le vulnerabilità vengano scoperte e corrette rapidamente. Si potrebbe ampliare molto il concetto, ma di base un software opensource non significa necessariamente che valga di meno di uno chiuso e a pagamento. **Buono.** Ragionamento maturo.

7 I package manager e gli app store sono simili perché centralizzano la distribuzione del software, permettono aggiornamenti automatici e verificano l'integrità dei pacchetti (ad esempio con firme digitali). Questo riduce il rischio di installare software malevolo o di corruzioni. **Corretto.** Centrato il tema fiducia/firme/centralizzazione

8 Il Content-Security-Policy (CSP) aiuta a difendersi principalmente da attacchi XSS, limitando le fonti da cui il browser può caricare ed eseguire script e altre risorse. **Giusto.** CSP = anti-XSS, risposta essenziale e precisa.

9 Un esempio di uso di POST invece di GET è l'invio di credenziali di login. POST è preferibile perché i dati non compaiono nell'URL e non vengono salvati nella cronologia del browser.**Perfetto**

10 Heartbleed era una minaccia grave perché permetteva a un attaccante di leggere porzioni di memoria dei server vulnerabili. In questo modo si potevano rubare informazioni sensibili come password, cookie di sessione e persino chiavi private SSL, compromettendo la sicurezza delle comunicazioni criptate.**Ottimo.** Spiegata la gravità reale di Heartbleed