REPORT:Password Cracking

Obiettivo e introduzione

L’obiettivo del report è quello di recuperare le password hashate nel database della DVWA e eseguire sessioni di cracking per recuperare la loro versione in chiaro. L'esercitazione mira a simulare un ciclo completo di attacco, dall'esfiltrazione dei dati sensibili (hash delle password), fino al cracking delle password per ottenerne il valore in chiaro.

Strategia sviluppata

L'attacco si articola in due fasi distinte e sequenziali:

1. **Fase Online (Esfiltrazione):** Sfruttamento della vulnerabilità nel campo "User ID" per iniettare codice **SQL** . L'obiettivo è "ingannare" il database affinché restituisca dati non previsti (tabella utenti e password) invece del solo nome utente.
2. **Fase Offline (Cracking):** Una volta ottenuti gli **hash**, si procede all'identificazione dell'algoritmo crittografico e al successivo attacco a dizionario per risalire alle password originali.

Esecuzione: SQL Injection

Analisi della Vulnerabilità

L'applicazione richiede un "**User ID**" in formato numerico (ad esempio, ID=1). Tuttavia, l'impostazione di sicurezza su "Low" comporta che l'applicazione non esegua la validazione del tipo di dato fornito e non utilizzi *Prepared Statements*. Questa vulnerabilità consente a un aggressore di alterare la query SQL originale.

Payload Utilizzato

‘UNION SELECT user, password FROM users

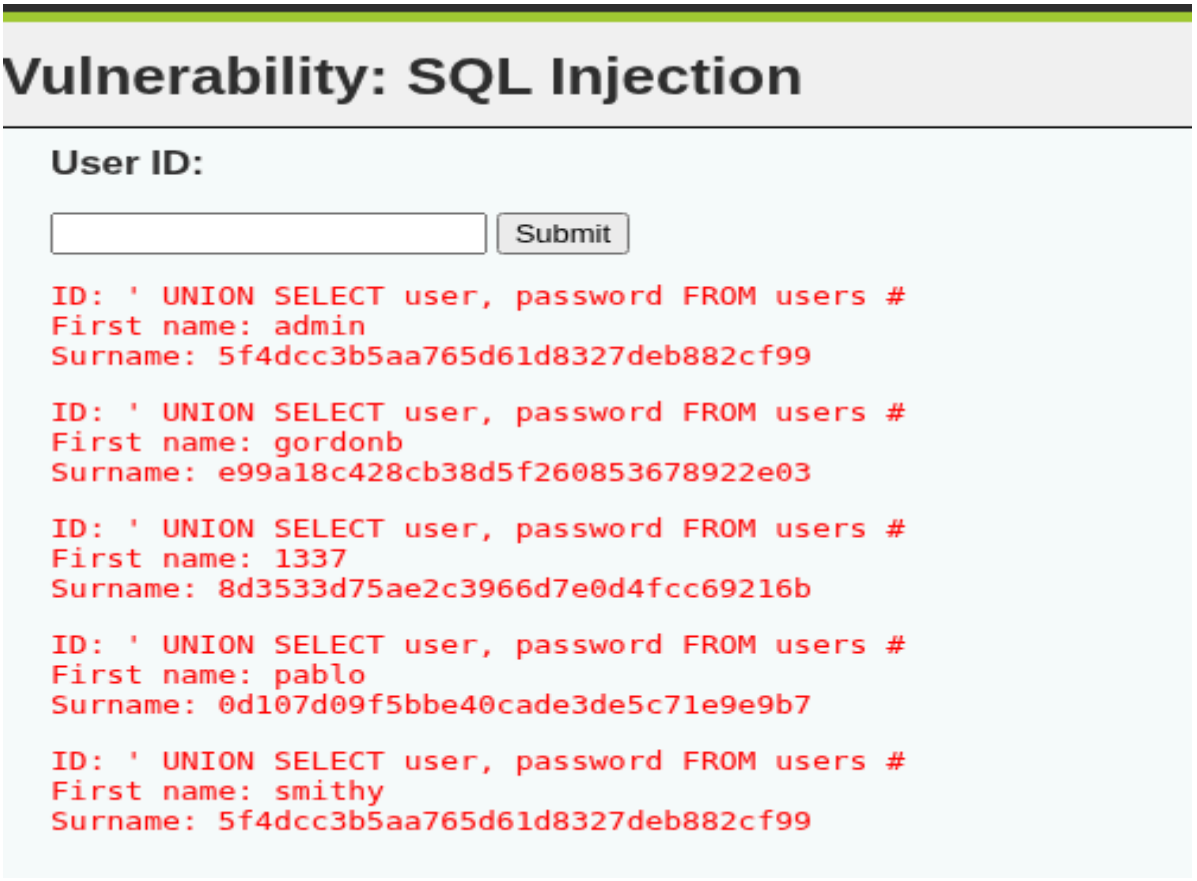
Approfondimento Tecnico del Payload:

- UNION: È l'operatore chiave dell'attacco. In SQL, **UNION** permette di combinare i risultati di due o più query **SELECT** in un unico set di risultati. Qui stiamo dicendo al DB: "Mostrami il risultato della tua ricerca ID... E UNISCI ad esso il risultato della mia richiesta (utenti e password)".
- SELECT user, password FROM users: La query malevola iniettata. Richiede esplicitamente l'estrazione delle colonne **user** e password dalla tabella sensibile **users** (nome tabella standard in molti DB).

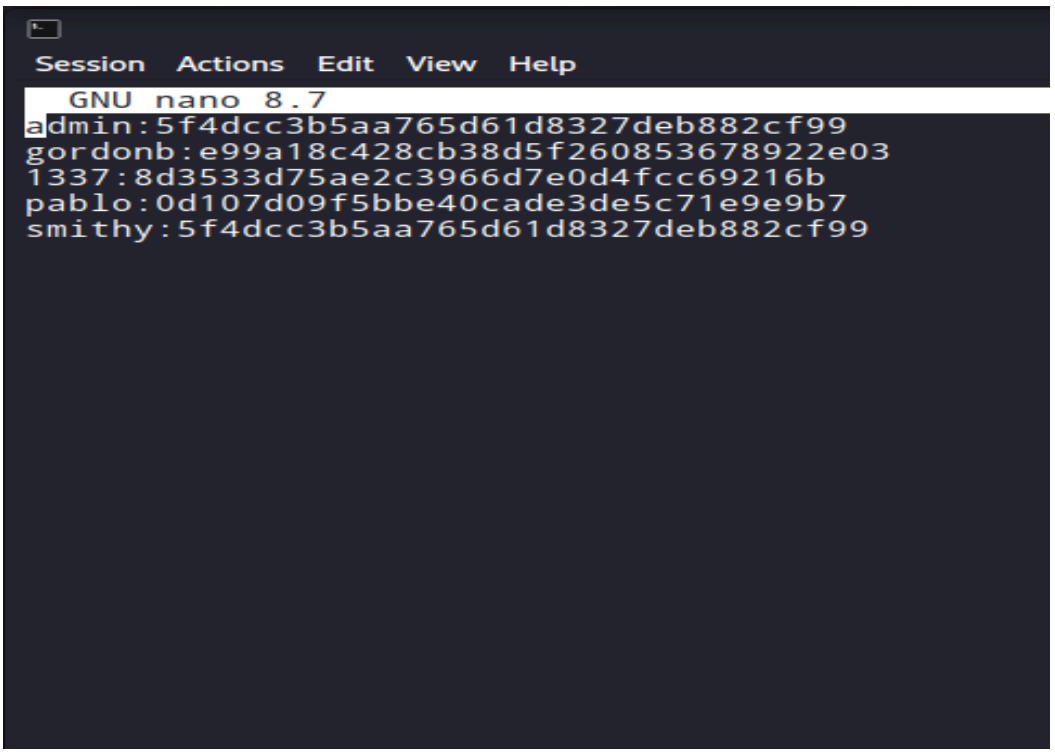
- #: Carattere di commento (sintassi specifica di MySQL). Questo simbolo neutralizza ("commenta") tutto il codice SQL successivo presente nella pagina originale, prevenendo errori di sintassi che altrimenti bloccherebbero l'esecuzione della query manipolata.

Risultato dell'Injection

L'iniezione ha avuto successo, restituendo a schermo l'elenco completo degli utenti associati ai rispettivi hash delle password, bypassando i controlli di accesso.



Le credenziali sono state salvate in un file **hash_pass_dwva.txt** per poter essere poi craccate da **John the Ripper.** Per farlo lanciamo il comando **nano hash_pass_dvwa.txt.**



Esecuzione: Password Cracking

Identificazione dell'Algoritmo

Prima di procedere al cracking, è stato necessario identificare la funzione di hash utilizzata.

L'analisi tramite il tool **hash-identifier** su una stringa campione di 32 caratteri (es. l'hash di admin) ha confermato l'utilizzo dell'algoritmo MD5.

L'**MD5** (Message Digest 5) è un algoritmo crittografico a 128 bit ormai obsoleto. È considerato insicuro per due motivi principali:

1. **Velocità:** È computazionalmente troppo veloce da calcolare. Questo è un vantaggio per l'attaccante, che può testare miliardi di combinazioni al secondo.
2. **Collisioni:** È matematicamente dimostrato che diverse stringhe possono generare lo stesso hash MD5.

Configurazione dell'Attacco (John the Ripper)

Per il cracking è stato utilizzato **John the Ripper (JtR)** . John the Ripper ("John") è uno dei software di password cracking (recupero password) più celebri e longevi nel mondo della cybersecurity. È uno strumento **offline**, ovvero lavora su file di hash rubati e non interagisce direttamente con il sito web vittima durante l'attacco, rendendolo molto silenzioso e difficile da rilevare dai sistemi di monitoraggio attivi.

Come Agisce (Meccanica di Funzionamento): A differenza di quanto si possa pensare, JtR non "decripta" la password (poiché l'hash è una funzione a senso unico non reversibile), ma opera per tentativi secondo il seguente ciclo logico:

1. **Generazione:** Prende una "Password Candidata" (es. una parola dal dizionario o una combinazione casuale).
2. **Hashing:** Applica alla candidata lo stesso algoritmo matematico usato dalla vittima (nel nostro caso, MD5).
3. **Confronto (Matching):** Confronta l'hash appena calcolato con l'hash rubato.
4. **Esito:** Se le stringhe sono identiche, la password è stata trovata. Altrimenti, passa alla candidata successiva.

Modalità di Attacco utilizzata: Dictionary Attack Per questo report è stata scelta la modalità "a Dizionario" (Dictionary Attack).

Nota sulla Wordlist: La **rockyou.txt** deriva da una reale violazione di dati avvenuta nel 2009 ai danni dell'azienda RockYou, esponendo oltre 14 milioni di password. È lo standard per testare la robustezza contro password comuni.

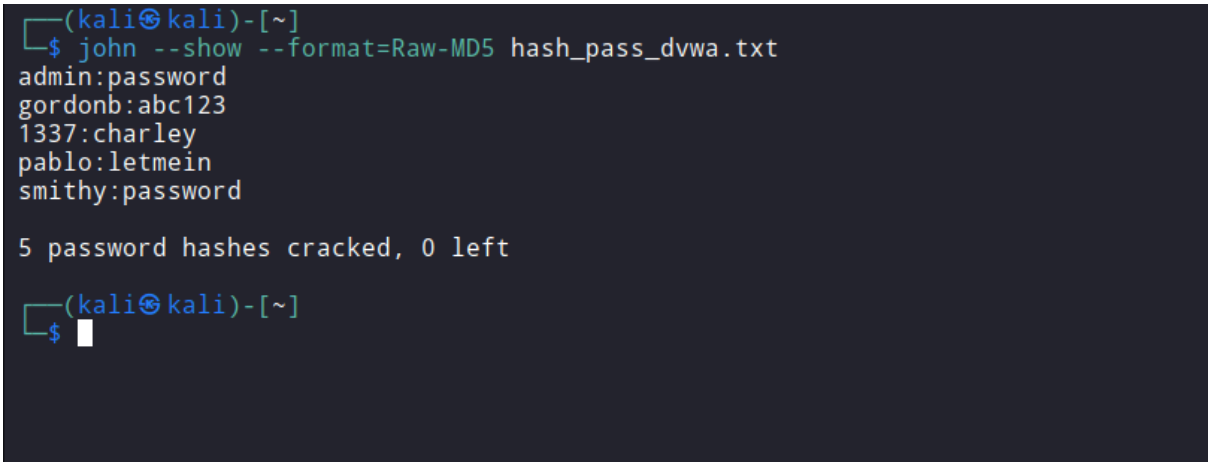
Comando eseguito:

Dettagli parametri:

- **--format=Raw-MD5**: Parametro cruciale. Quelli estratti dal DB sono hash "puri" o "grezzi" (Raw). Senza questa specifica, **JtR** potrebbe non riconoscere il formato corretto.
- **--wordlist**: Istruisce il software a non generare tentativi casuali (Brute Force puro), ma a testare le parole contenute nel dizionario indicato, riducendo drasticamente i tempi di esecuzione.

Risultati Ottenuti

L'attacco ha decifrato con successo le password in tempi estremamente ridotti (pochi secondi). Utilizzando il comando **john --show**, è stato possibile visualizzare le credenziali in chiaro:



Vulnerabilità emerse e conclusioni

L'esercizio ha messo in luce gravi vulnerabilità strutturali nel sistema target:

1. **Algoritmi di Hashing Deboli:** L'uso di **MD5** senza *Salt* (valore casuale aggiunto alla password prima dell'hashing) rende le password vulnerabili. Senza Salt, due utenti con la stessa password avrebbero lo stesso hash.
2. **Password Policy Inesistente:** Le password recuperate ("password", "123456", "letmein") dimostrano una totale assenza di requisiti minimi di complessità. Anche con algoritmi di hashing più robusti, password presenti in wordlist comuni come *rockyou.txt* verrebbero comunque compromesse rapidamente.

La simulazione dimostra che l'hashing non deve essere considerato solo come un metodo per "nascondere" la password, ma come un meccanismo per **guadagnare tempo** in caso di violazione. L'attuale configurazione (MD5 + Password deboli) non offre alcun margine di resistenza. La difesa necessaria richiede un approccio a due livelli: tecnico (adozione di algoritmi lenti e *salted* come **bcrypt** o **Argon2** per rendere il cracking computazionalmente costoso) e procedurale (imposizione di policy sulla lunghezza e complessità delle password per renderle immuni agli attacchi a dizionario).