VAPT sull'immagine Docker di OWASP Juice Shop

Sito ufficiale di Juice Shop: https://owasp.org/www-project-juice-shop/

Introduzione

Il presente report descrive l'attività di Vulnerability Assessment and Penetration Testing (VAPT) effettuata sull'immagine Docker di OWASP Juice Shop.

OWASP Juice Shop è un'applicazione web volutamente vulnerabile, utilizzata come piattaforma di addestramento per sviluppatori e professionisti della sicurezza informatica.

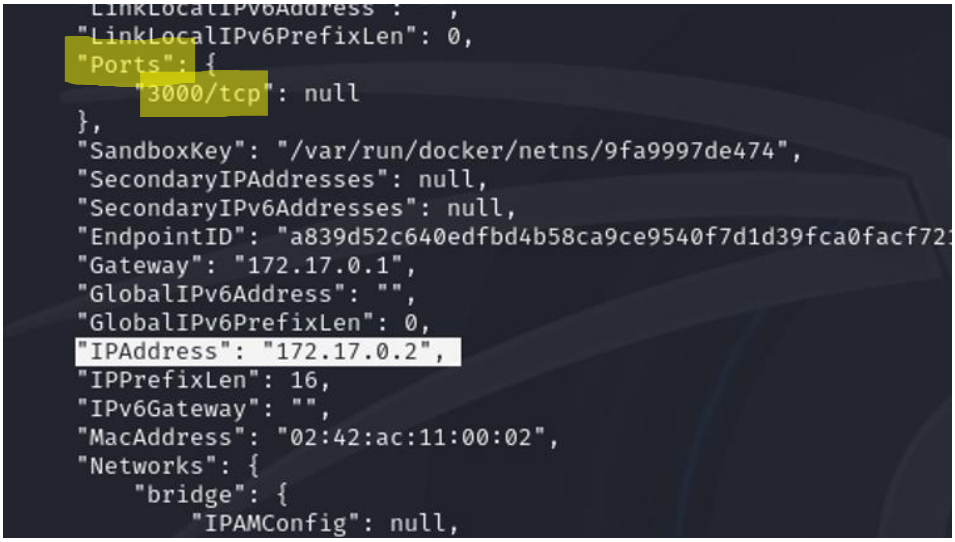
L'obiettivo principale del test è individuare e analizzare le vulnerabilità dell'immagine Docker, utilizzando una combinazione di strumenti automatizzati e tecniche di penetration testing per poi eventualmente proporre soluzioni per le vulnerabilità rilevate.

Per effettuare il VAPT abbiamo utilizzato una macchina virtuale con Kali Linux e una distribuzione Linux basata su Debian. Durante l'intero processo, è stato utilizzato un file di testo (.txt) detto "comandi progetto" per poter tenere traccia delle operazioni eseguite.

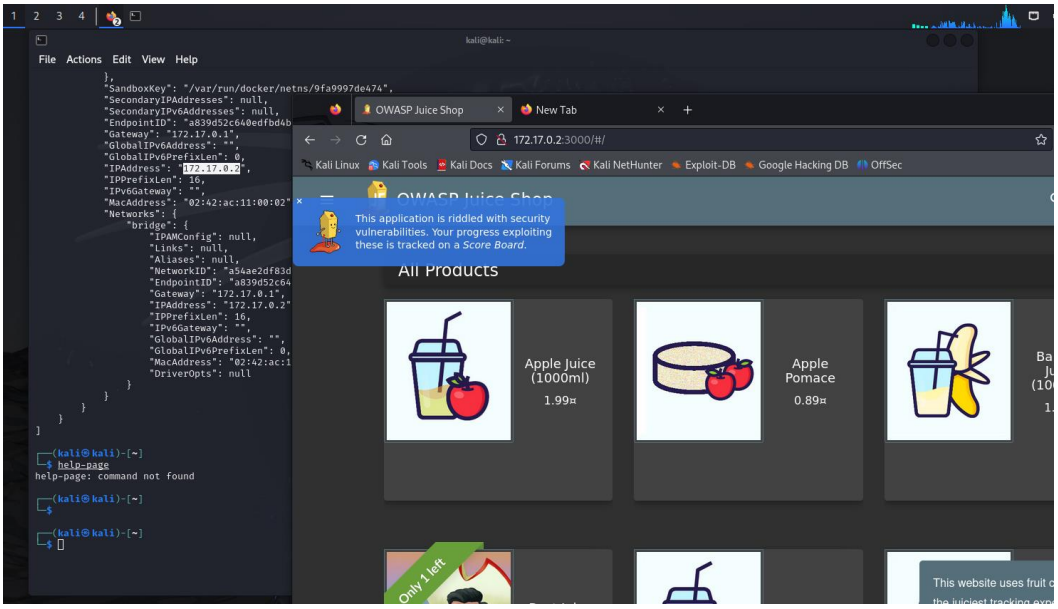
Prima di procedere con l'attività vera e propria, dopo aver installato Docker su Kali con una serie di operazioni, abbiamo scaricato ed eseguito l'immagine Docker utilizzando i comandi:

- docker pull bkimminich/juice-shop : per scaricare l'immagine Docker juice shop
- docker run -d bkimminich/juice-shop : per eseguire l'immagine

Analizzando l'immagine Juice Shop con il comando docker inspect b0dbca22f8a3 (che è l'ID del container che contiene l'immagine) otteniamo delle informazioni cruciali: indirizzo IP e porta.



Apriamo ora l'applicazione web



Azioni preliminari

Prima di iniziare con l'Information Gathering abbiamo creato una cartella condivisa sulla macchina virtuale nella quale verranno salvati tutti gli output e immagini relativi alle varie fasi del VAPT suddividendoli in sottocartelle. Per fare ciò abbiamo aperto Virtual Box, cliccato il tasto destro sull'icona della macchina virtuale, scelta opzione "setting", poi "shared folders", "new", per poi selezionare la cartella creata precedentemente sulla macchina locale selezionando gli ultimi due flag che permettono di mantenere aggiornata la cartella locale con quella virtuale.

Di seguito un elenco veloce di comandi usati:

- sudo mkdir /mnt/cartella-progetto-comandi = creazione della cartella principale
- sudo mount -t vboxsf cartella-progetto-comandi /mnt/cartella-progetto-comandi = per fare il mount della cartella locale con quella sulla macchina virtuale

1. Information Gathering

Per "information gathering" si intende la prima fase del penetration testing in cui si raccolgono informazioni rilevanti su un target (un sistema, una rete, un'applicazione, ecc.) che un tester di penetrazione può utilizzare per pianificare e condurre attacchi più efficaci.

1.1 Scansioni generali

Nella cartella denominata "nmapscansionebase" sono presenti tre file txt, ognuno relativo ad una breve scansione effettuata con Nmap, uno strumento open-source per la network exploration e l'auditing, progettato per scansionare rapidamente reti di grandi dimensioni (indicato anche per l'utilizzo verso singoli host).

- File "**output-tcp-docker.txt**" = ottenuto con il comando sudo nping -c 1 --tcp -p 3000 172.17.0.2 il quale invia un singolo pacchetto TCP alla porta 3000 dell'host con indirizzo IP 172.17.0.2. Questo può essere utilizzato per vari scopi, come ad esempio: testare la connettività verificando se l'host 172.17.0.2 è raggiungibile sulla porta 3000, fare una diagnostica di rete per aiutare a diagnosticare problemi di connettività di rete o di firewall, monitorare i servizi controllando se un servizio specifico in ascolto sulla porta 3000 è attivo. Nel file è possibile comprendere che:
  - Nping ha eseguito con successo un ping TCP tra il tuo host (172.17.0.1) e un altro host (172.17.0.2)
  - il pacchetto inviato ha ricevuto una risposta SYN-ACK, confermando che la porta di destinazione (3000) è aperta e pronta a ricevere connessioni TCP
  - i tempi di round-trip sono stati molto rapidi, indicando una connessione di rete efficiente tra i due host
- File "**output-nmap-aggressive.txt**" = analisi "aggressiva" attraverso il flag –A (comando completo: nmap -A 172.17.0.2); essa include version detecrion, OS detection e script scanning per avere una scansione completa e dettagliata del sistema bersaglio, compresa la versione del software, il sistema operativo e ulteriori informazioni che possono essere rilevanti per l'analisi delle vulnerabilità e la preparazione degli attacchi. Nel file è possibile comprendere che il dispositivo ospita OWASP Juice Shop su una porta aperta (3000/tcp), fornendo dettagli utili per capire come l'applicazione è configurata e quali misure di sicurezza sono in atto.
- File "**output-nmap.txt**" = ottenuto con il comando nmap 172.17.0.2, permette una scansione generica delle porte e dei relativi servizi attivi.

Durante la fase di Information Gathering sono stati utilizzati vari comandi e strumenti per raccogliere informazioni dettagliate sull'immagine Docker di OWASP Juice Shop. Di seguito, un riassunto del lavoro eseguito con i relativi comandi e le analisi effettuate.

1. **Scansione Nmap**, strumento utilizzato per il port scanning su un certo IP
   Comando utilizzato: sudo nmap -p 3000 -sV -T4 scanme.org > /mnt/cartella-progetto comandi/infogathering/output-gathering.txt
   Analisi:
   Lo stato della porta 3000 è risultato "filtered", il che significa che non è stato possibile stabilire una connessione a causa di un possibile filtraggio da parte di un firewall o di altre misure di sicurezza di rete. Questo suggerisce che la porta 3000 potrebbe essere soggetta a restrizioni di accesso.
2. **Uso di uno script automatizzato (auto_gathering.sh)**
   È stato creato uno script bash per automatizzare il processo di raccolta delle informazioni utilizzando vari strumenti di analisi. Lo script è stato salvato con il nome “auto_gathering.sh” e l’output generato è stato memorizzato in un file testo chiamato “autogathering.txt”.

Contenuto dello Script:

```
#!/bin/bash

IP="172.17.0.2"

PORT="3000"

echo "Ping Test"

ping -c 4 $IP

echo "Nmap Scan"

nmap -p $PORT $IP

echo "Banner Grabbing"

nc -v $IP $PORT

echo "Nikto Scan"

nikto -h http://$IP:$PORT

echo "Curl Headers"

curl -I http://$IP:$PORT

echo "WhatWeb Scan"

whatweb http://$IP:$PORT

echo "Dirb Scan"
```

dirb http://$IP:$PORT

Comando per eseguire lo script: ./auto_gathering.sh > /mnt/cartella-progetto comandi/infogathering/autogathering.txt

Analisi degli output:

2.1. Ping Test:
Verifica la raggiungibilità dell'host 172.17.0.2 inviando quattro pacchetti ICMP, i quali sono stati ricevuti con successo, confermando che l'host è raggiungibile e connesso alla rete.

2.2. Nmap scan:

Esegue una scansione della porta specificata, ossia la 3000/tcp. La porta è aperta e identificata come servizio "ppp". Questo suggerisce che un'applicazione sta ascoltando su quella porta, anche se il servizio "ppp" (Point-to-Point Protocol) potrebbe non essere corretto. Ciò può accadere se Nmap non riesce a determinare esattamente il servizio in esecuzione sulla porta.

2.3. Banner Grabbing:

Utilizza Netcat (nc) per recuperare il banner del servizio in esecuzione sulla porta 3000. Questo è uno strumento a riga di comando open source per la comunicazione remota su reti TCP/IP. Può essere utilizzato sia come client che come server e offre una vasta gamma di funzionalità per la gestione delle connessioni di rete, il trasferimento di dati e l'esplorazione di servizi di rete.

Il tentativo di banner grabbing non ha fornito informazioni utili sul servizio in esecuzione sulla porta 3000 a causa del timeout della richiesta (status code 408). Tuttavia, il fatto di aver ricevuto una risposta HTTP suggerisce che potrebbe esserci un server web o un servizio HTTP in esecuzione sulla porta.

2.4. Nikto Scan:

Utilizza Nikto, uno scanner per server web open source che esegue la scansione delle vulnerabilità dei server web. Questo strumento identifica potenziali vulnerabilità e problemi di configurazione nel web server.

Analisi e deduzioni:

1. **Informazioni Generali:**
   - Target IP: 172.17.0.2
   - Target Port: 3000
   - Server: Nessun banner recuperato, il che significa che Nikto non è riuscito a identificare direttamente il tipo di server web in esecuzione.

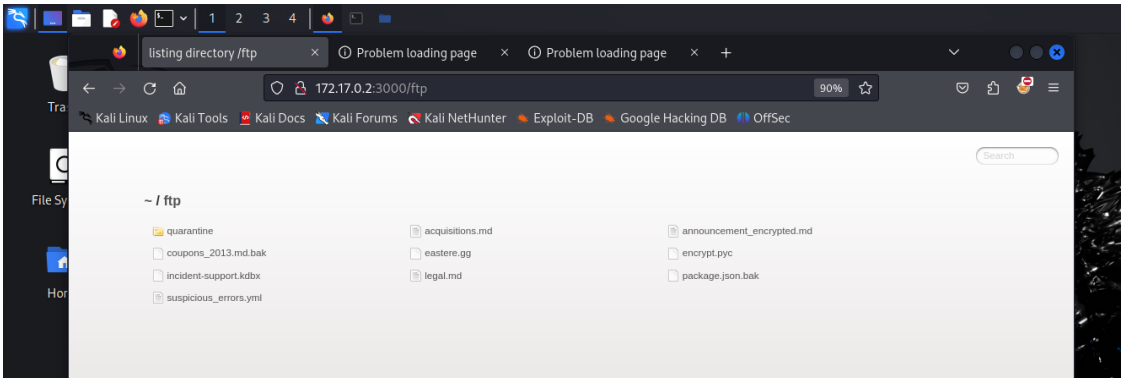
1. **Headers HTTP:**

    - Access-Control-Allow-Origin: Il valore * indica che l'accesso è consentito da qualsiasi origine, il che può avere implicazioni di sicurezza in alcuni contesti, specialmente se non configurato correttamente.
    - X-Recruiting: Header non comune trovato, con il contenuto `/#/jobs`. Questo suggerisce che il sito potrebbe avere una sezione dedicata alle opportunità di lavoro, accessibile tramite l'URL `/#/jobs`.
2. **File e Directory Identificati:**
    - robots.txt:
        - Contiene l'entry `/ftp/`, che ritorna un codice HTTP 200, indicando che questa directory esiste ed è accessibile. Questo potrebbe essere un punto di interesse per ulteriori test, in quanto potrebbe contenere file sensibili o utili.
        - Il file `robots.txt` deve essere esaminato manualmente per determinare quali altre direttive o directory possono essere presenti.
    - archive.cer: Un file di backup o certificato potenzialmente interessante è stato trovato. Questo potrebbe contenere informazioni sensibili che potrebbero essere utilizzate per ulteriori exploit.
3. **Headers di Sicurezza:**
    - X-Content-Type-Options: Questo header non è impostato. L'assenza di questo header può permettere al browser di rendere il contenuto del sito in modo diverso dal tipo MIME dichiarato, il che potrebbe portare a vulnerabilità come il MIME type sniffing.

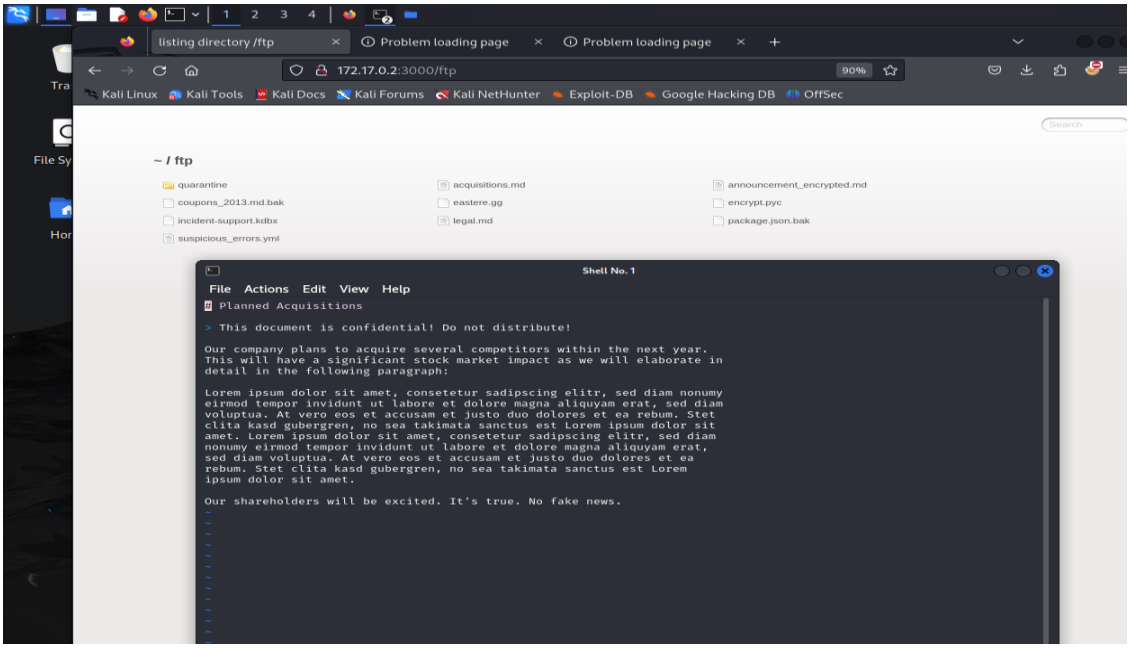
Riassumendo:

Nikto ha rivelato diverse potenziali vulnerabilità e aree di interesse per ulteriori indagini:

- **Access-Control-Allow-Origin:** Il valore * potrebbe esporre l'applicazione a rischi di sicurezza se non gestito correttamente.
- **robots.txt:** Contiene directory potenzialmente sensibili come `/ftp/` che sono accessibili e devono essere esaminate più a fondo.
- **Header di Sicurezza Mancanti:** L'assenza dell'header `X-Content-Type-Options` potrebbe esporre l'applicazione a rischi di sicurezza.
- **File di Backup:** Il file `archive.cer` potrebbe contenere informazioni sensibili che potrebbero essere sfruttate.



Se apriamo un file contenuto nella cartella “ftp” troviamo:



Ciò conferma la presenza di file ritenuti riservati, confidenziali.

2.5. Curl Headers:

Utilizza Curl per recuperare gli header HTTP della risorsa. Ciò che lo strumento fa è inviare una richiesta HTTP alla risorsa e ricevere in risposta solo le intestazioni HTTP, senza scaricare il corpo della pagina web o altri dati. Curl dunque funge da client HTTP e stabilisce una connessione con il server web specificato dall'indirizzo IP. Invia una richiesta HEAD (un metodo HTTP specifico per recuperare solo le intestazioni) e il server web risponde con un messaggio contenente le intestazioni HTTP relative alla risorsa richiesta.

Di seguito analizziamo l’output ottenuto, ossia gli headers HTTP e i relaviti dettagli:

Analisi e deduzioni:

4. **Stato HTTP:**

- HTTP/1.1 200 OK: La richiesta è stata elaborata con successo e la risposta contiene il contenuto richiesto.

5. **Headers di Sicurezza:**
    - Access-Control-Allow-Origin: *: Questo header indica che le richieste da qualsiasi origine sono permesse, il che può avere implicazioni di sicurezza se l'applicazione non gestisce correttamente i dati provenienti da fonti diverse.
    - X-Content-Type-Options: nosniff: Questo header impedisce ai browser di eseguire il "sniffing" del tipo MIME, proteggendo contro alcuni tipi di attacchi basati sul tipo di contenuto.
    - X-Frame-Options: SAMEORIGIN: Questo header impedisce il framing del contenuto del sito da parte di altri siti, proteggendo contro attacchi di clickjacking.
    - Feature-Policy: payment 'self': Questo header limita l'uso dell'API di pagamento al contesto corrente, migliorando la sicurezza contro abusi delle funzionalità del browser.
6. **Header Informativo:**
    - X-Recruiting: /#/jobs: Un header non comune che indica la presenza di una sezione di reclutamento accessibile tramite l'URL `/#/jobs`.
7. **Caching e Ottimizzazione:**
    - Cache-Control: public, max-age=0: Questo header permette il caching del contenuto, ma indica che deve essere sempre verificato per vedere se è stato modificato.
    - Last-Modified: Tue, 28 May 2024 12:59:27 GMT: Indica l'ultima modifica del contenuto, utile per la gestione della cache.
    - ETag: W/"ea4-18fbf48c07f": Un identificatore univoco per la versione del contenuto, utilizzato per la gestione della cache.
    - Accept-Ranges: bytes: Indica che il server supporta le richieste di range, permettendo di scaricare parti del contenuto.
8. **Tipologia e Lunghezza del Contenuto:**
    - Content-Type: text/html; charset=UTF-8: Il contenuto della risposta è HTML con codifica UTF-8.
    - Content-Length: 3748: La lunghezza del contenuto è di 3748 byte.
9. **Connessione:**
    - Connection: keep-alive: La connessione viene mantenuta aperta per ulteriori richieste, migliorando l'efficienza della comunicazione.
    - Keep-Alive: timeout=5: La connessione verrà mantenuta aperta per 5 secondi in attesa di ulteriori richieste.

Conclusioni:

I risultati della richiesta Curl confermano che l'applicazione web implementa diverse misure di sicurezza attraverso i suoi headers HTTP, come `X-Content-Type-Options` e `X-`

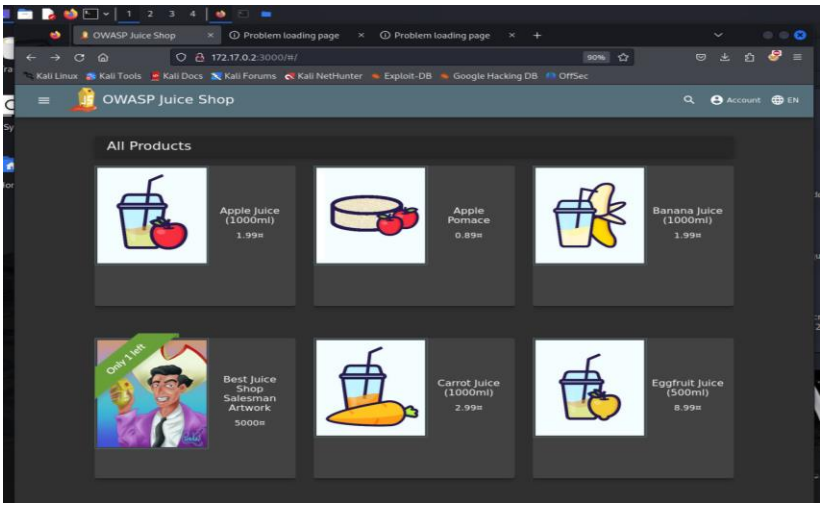
`Frame-Options`, che proteggono contro specifici tipi di attacchi. Tuttavia, l'uso di `Access-Control-Allow-Origin: *` richiede attenzione per assicurarsi che non esponga l'applicazione a rischi di sicurezza. La presenza di header come `X-Recruiting` suggerisce funzionalità aggiuntive che potrebbero essere esplorate ulteriormente. Inoltre, headers relativi alla cache e alla connessione indicano una configurazione volta a migliorare l'efficienza e le performance del sito.

2.6. WhatWeb Scan:

Utilizza WhatWeb per identificare le tecnologie web in uso. Questo è uno strumento di scansione web progettato per identificare le tecnologie software che compongono un sito web. Funziona analizzando il codice sorgente HTML, le richieste HTTP e le risposte, i file JavaScript e altri elementi di un sito web per creare un profilo dettagliato delle tecnologie utilizzate.

Analisi e deduzioni:

10. **URL Scansionato:**
    - URL: " http://172.17.0.2:3000 "
    - Stato della Richiesta: 200 OK, indicando che la richiesta è stata elaborata con successo e il contenuto è stato restituito correttamente.
11. **Informazioni sul Sito:**
    - Paese: RESERVED (riservato), con codice ZZ. Questo potrebbe indicare che l'indirizzo IP è stato assegnato, ma il paese non è stato specificato.
    - HTML5: Il sito utilizza HTML5, il che indica un moderno standard di markup.
    - Indirizzo IP: 172.17.0.2, che corrisponde all'IP del server.
    - JQuery: Versione 2.2.4 di JQuery è utilizzata sul sito.
    - Script: Il sito utilizza un modulo di script.
    - Titolo: Il titolo della pagina è "OWASP Juice Shop".
    - Header Non Comuni: Il sito include header non comuni come `access-control-allow-origin`, `x-content-type-options`, `feature-policy` e `x-recruiting`.
    - X-Frame-Options: L'opzione X-Frame è impostata su SAMEORIGIN, il che impedisce il framing del contenuto del sito da parte di altri siti.



Conclusioni:

Lo scan WhatWeb fornisce una panoramica delle caratteristiche e delle tecnologie utilizzate sul sito web. Il titolo della pagina suggerisce che si tratta dell'applicazione OWASP Juice Shop, confermando l'obiettivo del test di VAPT. Osserviamo, tra le verie informazioni di questa scannerizzazione, che l'applicazione utilizza la libreria JavaScript JQuery, ciò potrebbe tornare utile nella fase di Exploit per sfruttare delle sue vulnerabilità.

2.7. Dirb Scan:

Utilizza Dirb, un tool open-source per scansioni web progettato per trovare directory e file nascosti su un server web. Dunque, elenca le directory e i file accessibili, potenzialmente utili per ulteriori fasi di testing. Un primo tentativo è fallito a causa di errori di connessione, perciò è stato rieseguito ottenendo un risultato migliore.

- **DOWNLOADED**: Il numero di richieste effettuate (4612).
- **FOUND**: Il numero di percorsi trovati (9).

Questi risultati indicano che ci sono varie risorse e potenziali punti di accesso che si possono esplorare ulteriormente per ottenere più informazioni. Ad esempio:

- Esaminare il contenuto di `robots.txt` per vedere se contiene percorsi che potrebbero essere di interesse.

CONCLUSIONI FASE INFORMATION GATHERING

La fase di Information Gathering ha permesso di raccogliere una serie di informazioni critiche sull'immagine Docker di OWASP Juice Shop.

Per riassumere, elenchiamo le informazioni principali fino ad ora comprese e che potrebbero rilevarsi importanti:

- Si tratta di un servizio web chiamato OWASP Juice Shop
- L'applicazione utilizza una specifica versione di JQuery
- Dall'analisi della cartella /ftp/ osserviamo la presenza di file ritenuti riservati, a cui dunque non avremmo dovuto poter accedere. Dall'analisi dei file abbiamo compreso che Juice Shop utilizza il framework di backend Express per la sua applicazione web

Gli strumenti e i comandi utilizzati hanno fornito una visione dettagliata della configurazione del server, delle tecnologie in uso e delle potenziali vulnerabilità. Questi dati saranno fondamentali per le fasi successive, permettendo di indirizzare meglio gli sforzi di analisi e sfruttamento delle vulnerabilità.

2. Vulnerability Assessment

Questa fase serve principalmente a identificare e valutare le vulnerabilità presenti nei sistemi, nelle reti o nelle applicazioni dell'organizzazione oggetto del test. Questa è una fase cruciale perché fornisce una panoramica dettagliata delle potenziali debolezze che potrebbero essere sfruttate da un attaccante per compromettere la sicurezza dell'ambiente testato. Per avere una visione più completa possibile, abbiamo deciso di usare Nikto, come nella fase precedente. Qui, però, ci siamo voluti concentrare solo sull'uso di questo specifico strumento; di seguito una breve spiegazione dei passi compiuti e dei dati ottenuti da essi.

2.1 Scansione Vulnerabilità con Nikto

Comando utilizzato: nmap -sV -p 3000 --script vuln 172.17.0.2 > /mnt/cartella-progetto-comandi/vulnasses/nikto_vuln/ nikto_scan_output.txt

L'output di questo comando viene salvato in un file denominato "vuln-scan.txt" all'interno della cartella "nikto_vuln". Di seguito è riportata una spiegazione dettagliata di ciascun punto rilevato dallo scan:

- **Header HTTP non comuni e configurazione delle intestazioni:**
  - `/`: Intestazione `access-control-allow-origin` impostata su `*`, che permette a qualsiasi origine di accedere alle risorse, potenzialmente rischioso per la sicurezza se non configurato correttamente.
  - `/`: Intestazione `x-recruiting` con il contenuto `/#/jobs`, non comune ma probabilmente non pericolosa.
  - Nessuna intestazione `X-Content-Type-Options` trovata, che potrebbe consentire a un agente/utente di interpretare il contenuto in modo diverso dal tipo MIME dichiarato, aprendo la strada a potenziali attacchi di tipo MIME-sniffing.
- **File e directory potenzialmente sensibili**
  - `/robots.txt`: Contiene l'entry `/ftp/`, accessibile con codice HTTP 200, che potrebbe rivelare informazioni sensibili.

- Molti file di backup, certificati e archivi potenzialmente interessanti, come `archive.cer`, `2.jks`, `17217.cer`, ecc., che potrebbero contenere informazioni sensibili o chiavi di sicurezza.

- **Vulnerabilità e problematiche di sicurezza**
  - La presenza di file di backup e certificati potrebbe indicare una gestione inadeguata dei file sensibili, esponendo il sistema a potenziali attacchi se questi file vengono scaricati e analizzati dagli attaccanti.
  - L'assenza di alcune intestazioni di sicurezza (come `X-Content-Type-Options`) potrebbe rendere il server vulnerabile a specifici tipi di attacchi, come quelli di tipo MIME-sniffing.

Date le vulnerabilità appena spiegate, è possibile evitarle o risolverle attraverso delle best practice come: rimuovere o proteggere i file sensibili assicurandosi che i file di backup, i certificati e gli archivi non siano accessibili pubblicamente o, se necessario, eliminarli, aggiungere intestazioni di sicurezza come `X-Content-Type-Options` per mitigare i rischi associati al MIME-sniffing e rivedere e aggiornare il file robots.txt limitando le informazioni esposte nel file per non rivelare directory o risorse sensibili.

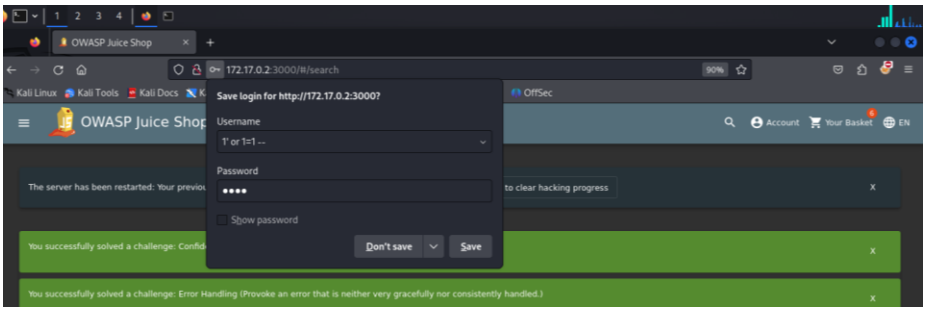
In generale, l'output di Nikto evidenzia diverse aree di miglioramento nella configurazione e gestione della sicurezza del server web. Implementare le misure suggerite contribuirà a ridurre le vulnerabilità e a proteggere meglio il sistema contro potenziali attacchi.

2.2 SQL Injection

SQL Injection è una vulnerabilità che consente ad un utente malintenzionato di interferire con le query che un'applicazione effettua sul proprio database per visualizzare dati che normalmente non è in grado di recuperare, come i dati appartenenti agli utenti o qualsiasi altro dato a cui l'applicazione può accedere. Inoltre è la prima della Top Ten OWASP, rappresentando dunque la vulnerabilità più critica di tutte.

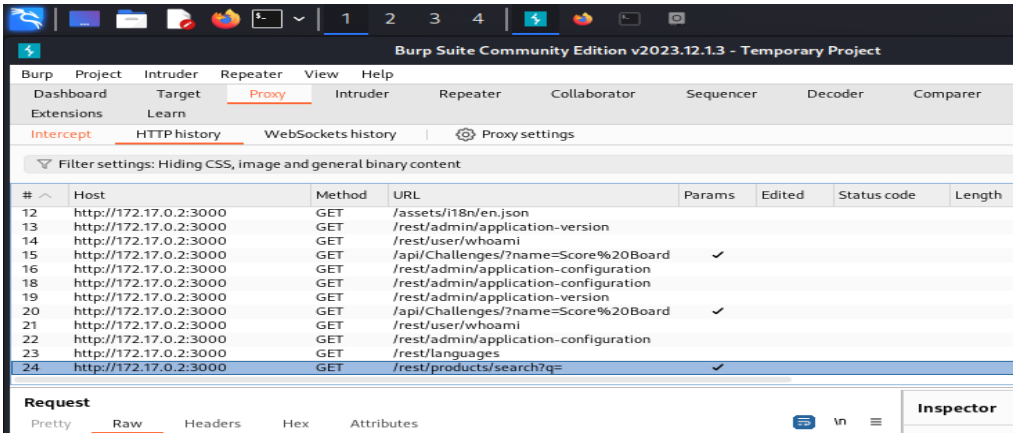
Per prima cosa, per testare se Juice Shop è vulnerabile a SQL Injection, è possibile utilizzare un insieme sistematico di test manualmente in ogni punto di input dell'applicazione. Abbiamo quindi provato ad eseguire un accesso ad un account senza conoscere username e password e senza averne creato uno. Per fare ciò, nella sezione Login dove sono presenti campi di input, abbiamo provato ad inserire delle condizioni come "1' or 1 = --" nel campo username e lettere a piacere nel campo password.

Premendo invio, eseguiamo il login con successo (come mostrato in figura qui sotto).

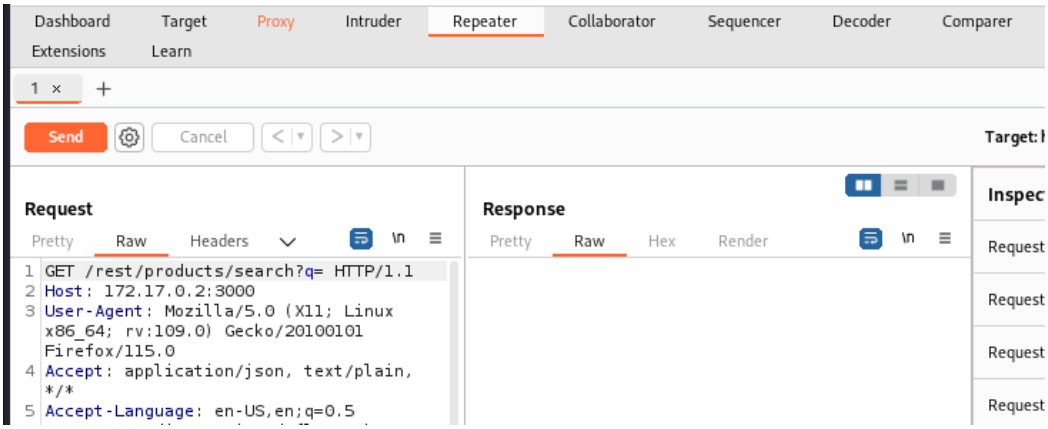


Possiamo quindi dedurre che l'applicazione è vulnerabile a SQL Injection.

A questo punto, utilizzando BurpSuite che funge da proxy, visualizziamo le richieste che vengono effettuate verso l'applicazione e analizziamole. Se proviamo a cercare in Juice Shop qualche elemento usando la barra di ricerca, effettueremo una richiesta, la quale sarà visibile su BurpSuite.



Cercando nella sezione HTTP history, troviamo infatti la richiesta precedentemente effettuata: /rest/products/search?q=. Adesso, premendo tasto destro sulla richiesta e inviandola al Repeater, visualizziamo una schermata più specifica in cui poter manipolare direttamente la query "q=" per ottenere dei dati.



3. Exploitation

L'exploitation (o sfruttamento), eseguito dopo l'identificazione di vulnerabilità, rappresenta una fase fondamentale e si pone come obiettivo quello di compromettere il sistema vulnerabile e svolgere attività malevole attraverso essa.

Per questa fase abbiamo deciso di usare SearchSploit, tool incluso in Kali Linux che facilita la ricerca di exploit mantenendo in locale una copia di questi attraverso il flag "-m" in una

directory specificata dall'utente. Infine, l'output ottenuto della scansione delle varie vulnerabilità note è stato salvato all'interno del file "jquery.txt" in una cartella denominata "exploitation".

3.1 DoS

Un attacco DoS (denial-of-service) è un attacco informatico in cui l'attaccante cerca di impedire agli utenti di accedere alle risorse di un servizio inviando intenzionalmente un elevato numero di richieste ad uno specifico target.

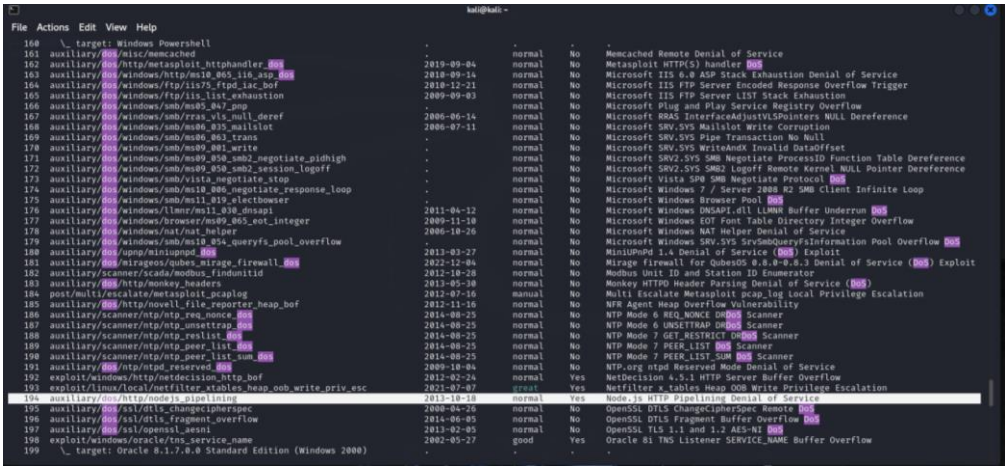
3.1 DoS con Metasploit

Metasploit è un framework open-source fornito da Kali per effettuare exlpoitation. Nello specifico, è stato usato per effettuare un attacco di tipo DoS su Juice Shop.

Di seguito i comandi passo per passo per effettuarlo:

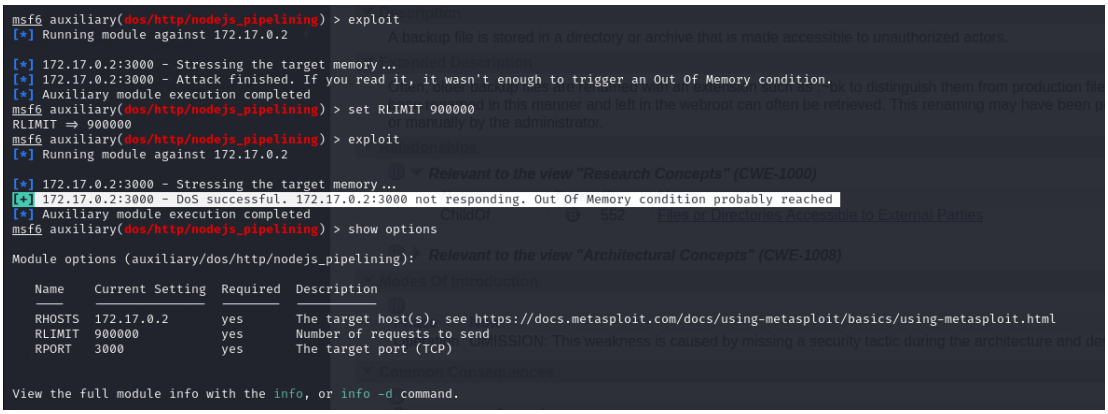
- msfconsole = per avviare il terminale di metasploit
- search dos = metodo di ricerca con parole chiave per identificare moduli vulnerabili che verranno successivamente utilizzati
- use auxiliary/dos/http/nodejs_pipelining = selezione del modulo più interessante per attuare l'attacco. Il server HTTP in Node.js 0.10.x prima di 0.10.21 e 0.8.x prima di 0.8.26 consente agli aggressori remoti di causare una negazione del servizio (consumo di memoria e CPU) inviando un gran numero di richieste in pipeline senza leggere la risposta (https://www.cvedetails.com/cve/CVE-2013-4450/)
- show options = mostrare informazioni di base relative a quel modulo (p.e. porta e host)
- set RHOST 172.17.0.2
  set RPORT 3000
  set RLIMIT 900000
  = con questa serie di comandi abbiamo modificato i valori preconfigurati per renderli conformi e utili ad effettuare l'attacco desiderato
- run = effettuiamo l'attacco vero e proprio; infatti, causa una condizione di esaurimento della memoria (Out Of Memory) sul target

Identificazione modulo auxiliary/dos/http/nodejs_pipelining nell'elenco di metasploit dopo aver utilizzato "search dos":



Usiamo il modulo. All'inizio si presenterà così e dovrà essere configurato:

Una volta cambiati i parametri come spiegato precedentemente, l'attacco andrà a buon fine come presentato di seguito:



Conclusioni:

Dato il successo dell'attacco DoS, si potrebbe dedurre che Juice Shop sia vulnerabile come servizio web. Non è necessariamente vero però che se questo tipo di attacco va a buon fine, allora il target colpito è vulnerabile. Infatti, questo attacco potrebbe avere successo anche contro applicazioni ritenute non vulnerabili per diverse ragioni come problemi di infrastruttura o errori di configurazioni del firewall o di altri componenti del sistema.

L’attacco di tipo DoS è estremamente pericoloso e dannoso per diversi motivi: può provocare l’interruzione del normale funzionamento di un sistema causando perdite finanziarie per le aziende che dipendono dai loro sistemi, può causare il danneggiamento della reputazione, facendo apparire un’organizzazione inaffidabile, dovendo poi subire costi elevati per poter ripristinare i sistemi, oppure, in casi più gravi, può essere utilizzato per impedire alle persone di accedere a servizi critici come servizi di emergenza.

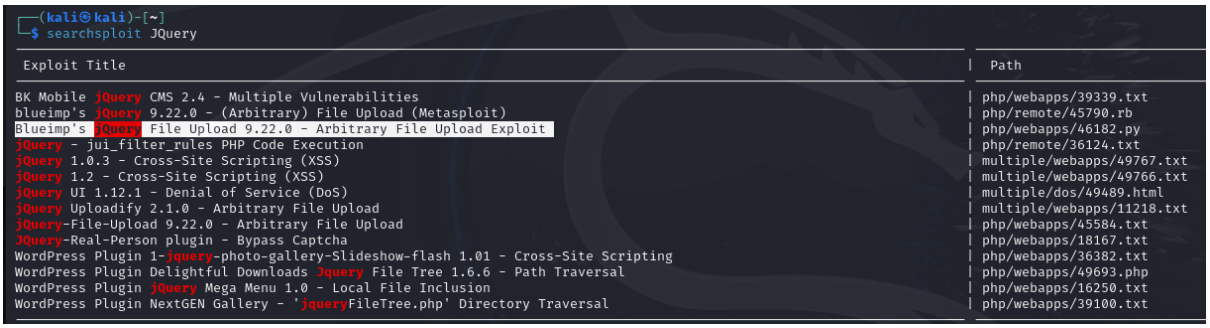
Esistono diverse misure di sicurezza che le organizzazioni possono implementare per ridurre il rischio di subire un attacco di questo tipo, come considerare l’implementazione di un modello di sicurezza Zero Trust, mettere in pratica la security by design, includere la sicurezza informatica nella continuità aziendale, nel ripristino di emergenza e nella pianificazione della risposta alle emergenze o ancora includere questo tipo di attacco, insieme alle sue varianti come DDoS, all’interno dei Penetration Test per simulare attacchi complessi, identificare vulnerabilità e rafforzare la difesa.

3.2 Arbitrary file upload exploit

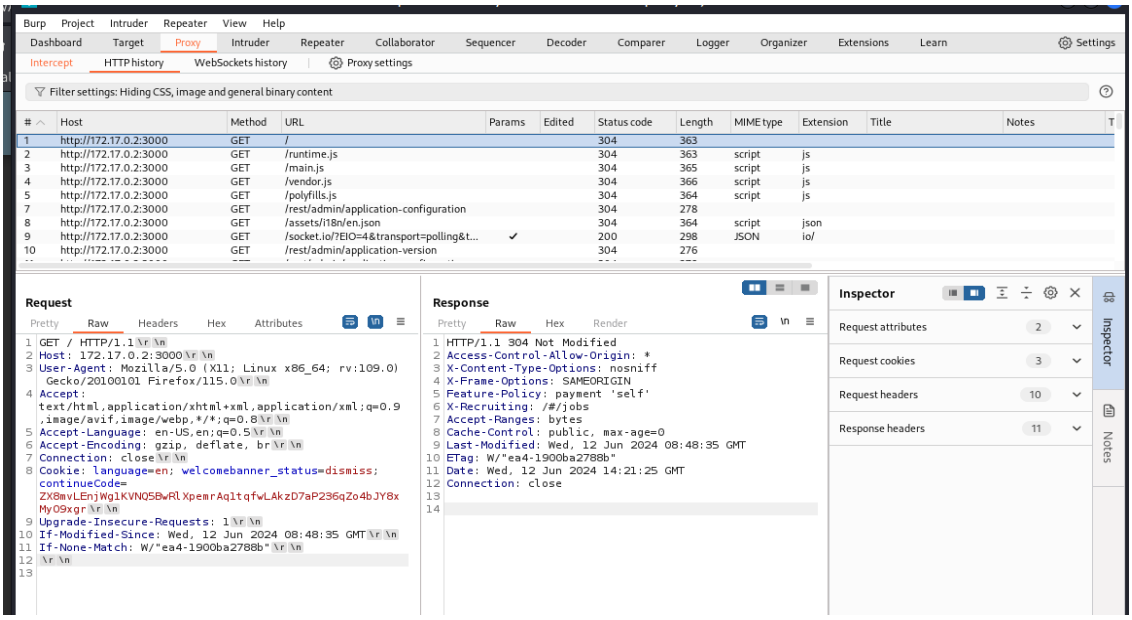
Comandi usati:

- searchsploit -m php/webapps/46182.py ls la vulnerabilità identificata dall’id 46182 sfrutta le carenze nei controlli di validazione o filtraggio dei file caricati in applicazioni web; quindi, una volta caricato, il file dannoso può essere utilizzato per eseguire codice arbitrario, accedere a dati sensibili o compromettere ulteriormente il sistema. Il comando, dunque, fa caricare un file senza che quest’ultimo venga rilevato durante il monitoraggio dei file.

Identificazione del modulo nell’elenco di searchsploit:

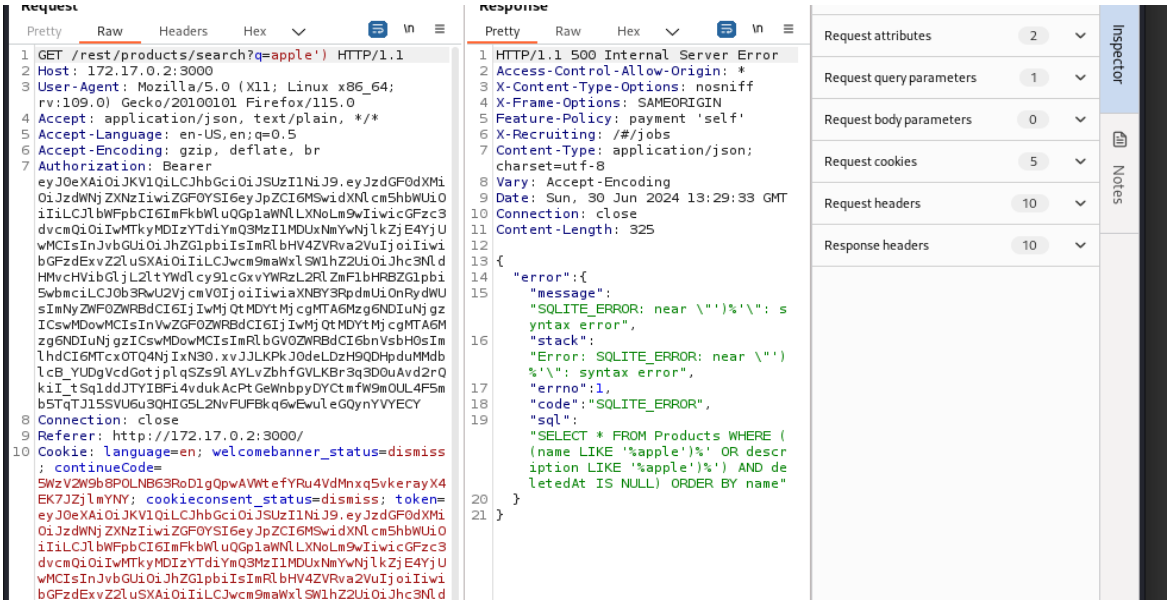


Dopo aver caricato il file, utilizzando Burp Suite, notiamo nessuna prova o traccia che l’upload sia avvenuto.

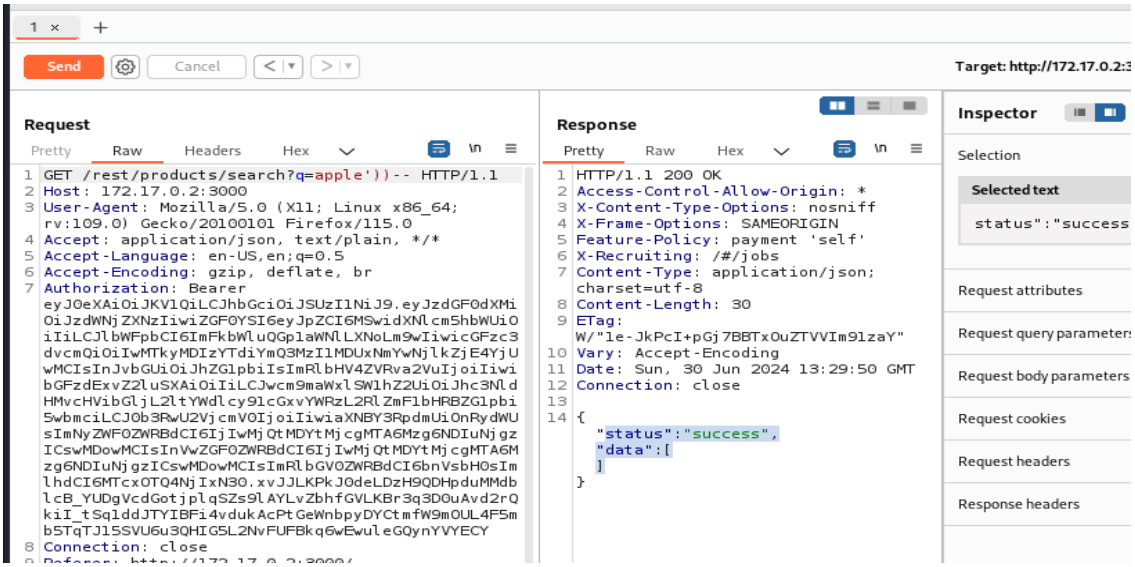


3.3 SQL Injection

Manipolando la query “q=” utilizzando BurpSuite, abbiamo innanzitutto ottenuto degli errori, dai quali si evince che l’applicazione utilizza SQLite, una libreria C che implementa un RDBMS (sistema di gestione di DB relazionali) SQL veloce, autonomo, serverless e completo.



Cambiando ulteriormente la query, otteniamo un messaggio di successo (success), confermando che Juice Shop è vulnerabile a SQL Injection.



4. Post Exploitation

Questa fase segue l'avvenuto exploit di un target system e ha come obiettivo quello di reperire informazioni sensibili e rilevanti per la sicurezza locale o del target business. Inoltre, è possibile che l'attaccante tenti di aumentare i propri privilegi all'interno del sistema compromesso per avere maggiore controllo, oppure che installi una backdoor per mantenere l'accesso al sistema dopo la fine del VAPT.

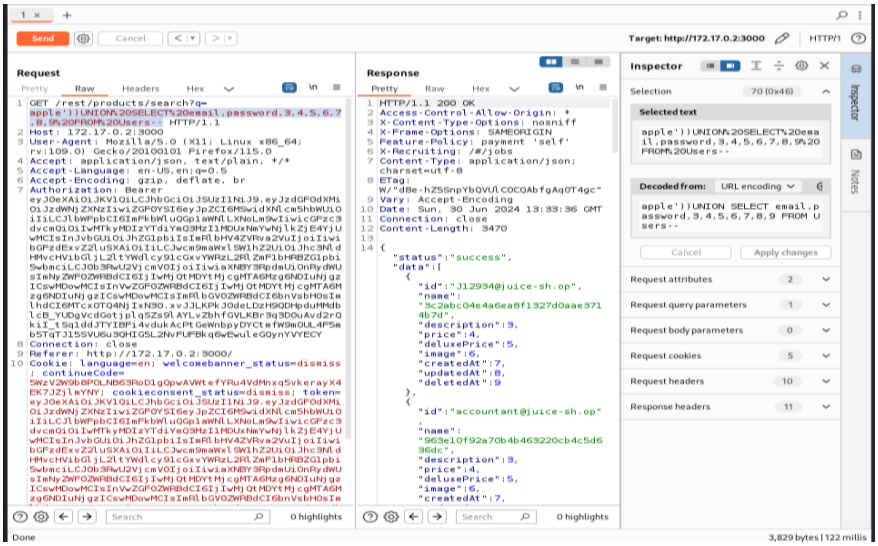
4.1 DoS

Eseguendo un attacco DoS su Juice Shop, non è stato possibile installare una backdoor per poter recuperare informazioni sensibili. Di certo, il fatto che l'attacco sia andato a buon fine, rappresenta un punto vulnerabile per l'applicazione.

Dato che in questo caso non sono stati reperiti ulteriori dati, proseguiremo con SQL Injection, dove invece è stato possibile.

4.2 SQL Injection

La query "q=", scritta precedentemente come "q=apple'))-- ", è stata nuovamente modificata per provare ad ottenere dati sensibili, come lo schema del database dell'applicazione contenente i dati degli utenti registrati. La query che abbiamo usato è: q=apple'))UNION%20SELECT%20email,password,3,4,5,6,7,8,9%20FROM%20Users-- . Nella condizione è stata usata la parola chiave UNION per eseguire una query SELECT aggiuntiva e aggiungere i risultati alla query originale. Ciò consente di recuperare dati da altre tabelle del database, formando un attacco noto come UNION SQL Injection.



Abbiamo così ottenuto la lista degli utenti registrati contenente dati sensibili come username (id) e password (name), sebbene queste ultime siano criptate.

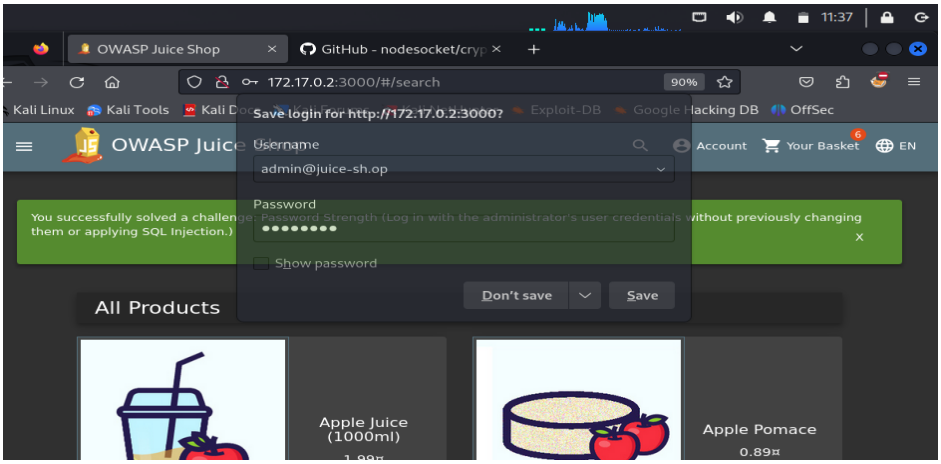
Conclusioni:

Ciò rappresenta una vulnerabilità molto grave, poiché i malintenzionati possono venire a conoscenza delle credenziali degli utenti attraverso pochi e semplici passaggi. Altre conseguenze che questa vulnerabilità può causare sono: compromettere l'integrità dei dati e la privacy degli utenti, esporre dati sensibili e concedere all'attaccante la possibilità di ottenere i permessi da amministratore.

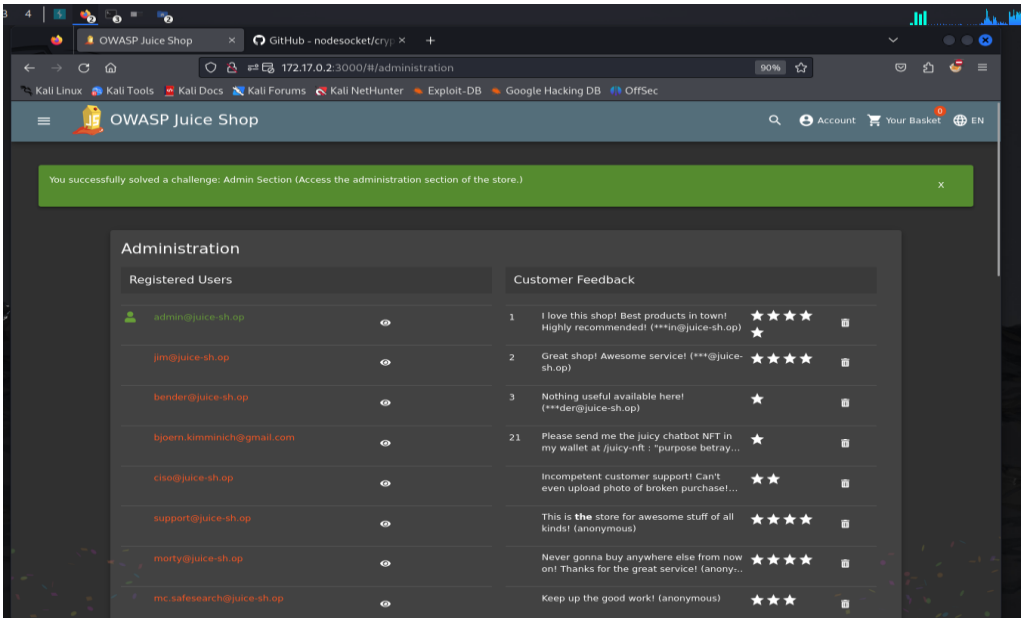
Quest'ultimo consiste in un fattore estremamente pericoloso, dato che è probabile sia presente un utente con privilegi di amministratore. Perciò, un aggressore potrebbe accedere poi al sistema apportando modifiche non autorizzate o utilizzando codice dannoso.

4.2.1 SQL Injection – admin login

Per testare se effettivamente fosse possibile accedere come amministratore, abbiamo provato ad eseguire l'accesso all'account dell'admin con l'email admin@juice-sh.op recuperata dallo schema del DB. Non essendo a conoscenza della password, abbiamo provato ad indovinare, ottenendo con successo un riscontro utilizzando "admin123", che è troppo semplice e molto debole, rendendo l'account poco sicuro.



Ottenendo così i permessi da amministratore, abbiamo cercato ulteriori informazioni sensibili, fino a trovare una lista di utenti (indirizzi e-mail) e i relativi commenti rilasciati all'interno dell'applicazione. Volendo, si possono eseguire operazioni come la rimozione di un commento, premendo l'icona accanto a destra.



Conclusioni, OWASP Top10

Siamo dunque riuscite a identificare diverse vulnerabilità che possono essere catalogate in base alla loro criticità grazie alla Top10 di OWASP:

1. Vulnerabilità "**injection**": situata al primo posto della Top10 e analizzata al punto "2.2 SQL Injection" della relazione
2. Al terzo posto, "**Sensitive Data Exposure**": vulnerabilità che consiste nel rivelare informazioni sensibili, come ad esempio la password, la quale è stata da noi rilevata e spiegata al punto 4.2 e 4.2.1

3. Vulnerabilità “**XML External Entities (XXE)**” al quarto posto della Top10: consiste nel poter caricare file esterni malevoli, e noi lo abbiamo riscontrato e descritto al punto “3.2 Arbitrary file upload“ della relazione
4. “**Broken Access Control**”: vulnerabilità di criticità 5/10 che, a causa di misure di sicurezza inadeguate, permette a utenti non autorizzati di accedere a risorse; nel nostro caso l’inadeguatezza è data dal fatto che vi è un solo metodo di login, mentre, se fosse stata una MFA (autenticazione a più fattori), sarebbe risultato più sicuro.