Sigurnost operacijskih sustava i aplikacija
Tin Plavec

Laboratorijska vježba 4
Izvještaj

Instalacija i pokretanje

OWASP Mutillidae

Aplikaciju Mutillidae najlakše je pokrenuti pomoću Docker kontejnera koristeći docker-compose. Nakon kloniranja repozitorija (https://github.com/webpwnized/mutillidae-docker), potrebno je ući u klonirani direktorij i izvršiti:

```
docker-compose up
```

Docker će povući potrebne slike te zavrtiti 5 kontejnera: *www*, *database*, *database_admin*, *ldap*, *ldap_admin*. Nama najvažniji je web poslužitelj *www*, koji je dostupan na http://localhost:80 .

Koristio sam mutillidae-docker verziju 1.0.39, koja koristi najnoviju Mutillidae II verziju (https://github.com/webpwnized/mutillidae, trenutno verzija 2.11.4).

OWASP ZAP

Zed Attack Proxy instalirao sam kao Linux paket (https://www.zaproxy.org/download/) na operacijski sustav Ubuntu 22. Nakon skidanja paketa, jednostavno se pokrene skripta `zap.sh` koja pokreće GUI aplikacije. Odabrao sam neperzistentnu sesiju te omogućio sve preporučene dodatke.

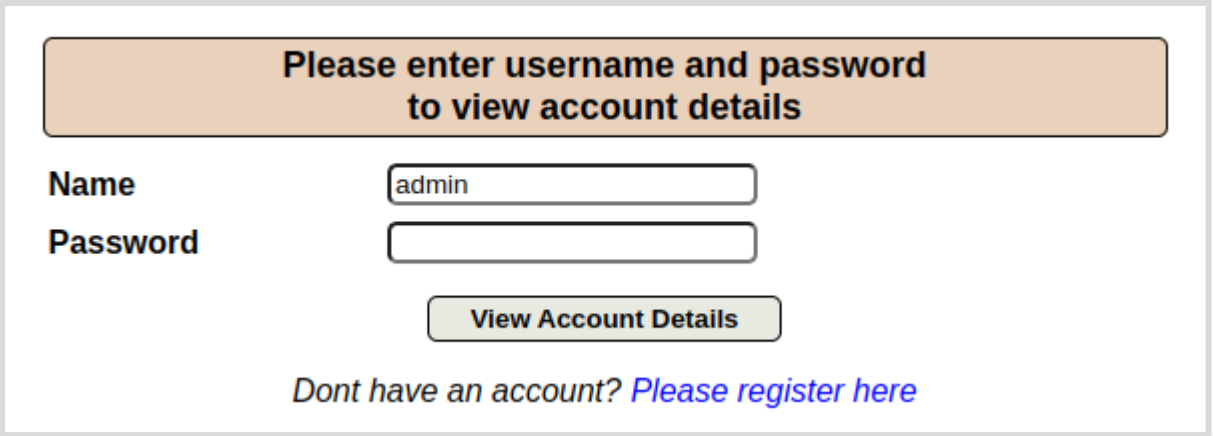
Koristio sam verziju 2.12.0 .

Neizrazito testiranje u ZAP-u

Prije konfiguracije neizrazitog testiranja, potrebno je napraviti jedan normalan zahtjev kroz ZA proxy. Na početnom zaslonu, odabrat ćemo `Manual Explore`, te unijeti `http://localhost` obzirom da je tamo dostupna Mutillidae aplikacija. Zatim ćemo označiti `Enable HUD` ako već nije, te pokrenuti Chrome. Otvorit će se novi prozor preglednika, koji automatski koristi ZAP-ov proxy.

SQL Injection

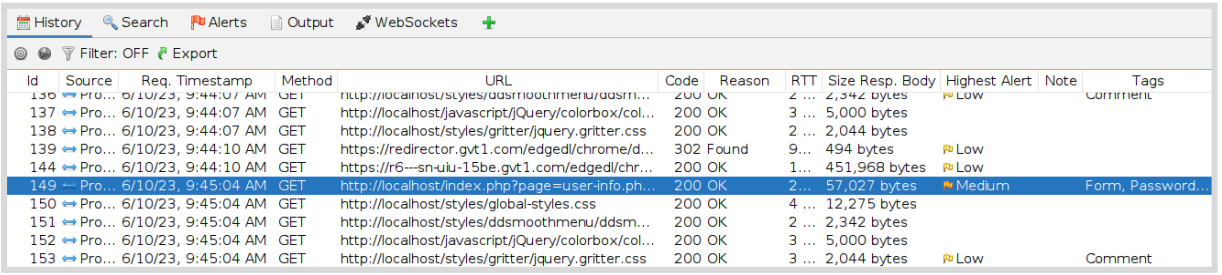
U novootvorenom prozoru, obrat ćemo OWASP 2017 > A1 - Injection (SQL) > SQLi Extract Data > User Info.



Za `name` ćemo unijeti `admin` te izvršiti zahtjev. Aplikacija će nam odgovoriti:

```
Results for "admin".0 records found.
```

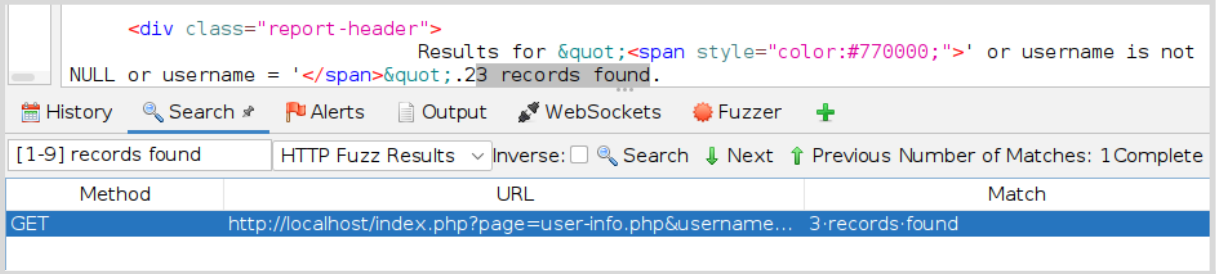
Nakon što smo izvršili zahtjev, možemo zatvoriti ovaj prozor. U ZAP-u ćemo pogledati povijest zahtjeva, gdje bi trebao biti i naš zahtjev kojeg ćemo odabrati:



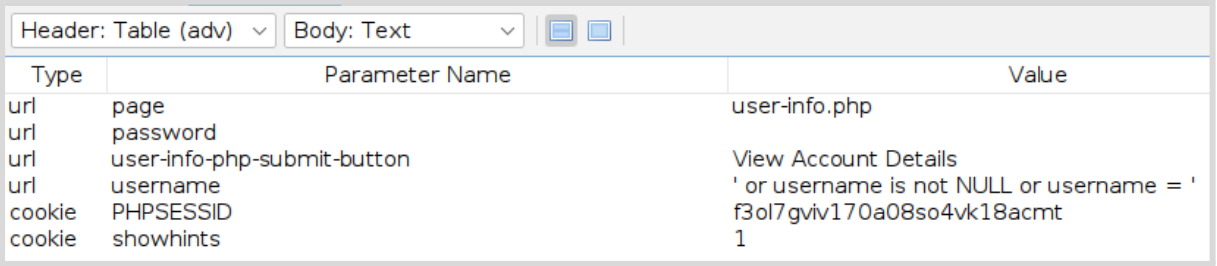

```
GET
http://localhost/index.php?page=user-info.php&username=admin&password=&user-info-php-submit-button
=View+Account+Details HTTP/1.1
Host: localhost
Connection: keep-alive
sec-ch-ua: "Not.A/Brand";v="8", "Chromium";v="114", "Google Chrome";v="114"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.
0 Safari/537.36
```

Primijetimo parametre zahtjeva: `&username=admin&password=`
Cilj neizrazitog testiranja je umjesto očekivanih unosa, isprobati razne neočekivane unose. U ovom slučaju dovoljno će biti testirati `username` parametar. Cilj nam je isprobati klasične SQL Injection napade, za što nam je dovoljno jedno polje. Stoga ćemo označiti `admin`, kliknuti desni klik, te odabrati `Fuzz…`. Otvorio se novi prozor u kojem bi `admin` trebalo biti također označeno. Sada trebamo odabrati što ćemo unijeti umjesto niza `admin` klikom na `Payloads…`. Dodat ćemo *payload* tipa `File Fuzzers`. Odaberimo jbrofuzz > Injection, te sve 3 MySQL stavke. Prilikom odabira možemo pogledati što će se točno unijeti umjesto `admin`. Svaki redak predstavlja jedan korak testiranja. U ovom slučaju aplikacija će se testirati pomoću čestih izraza za SQL Injection napad.
Pokrenimo sada testiranje pomoću gumba `Start Fuzzer`. Testiranje bi trebalo završiti za nekoliko sekundi. Sada možemo pretraživati rezultate. Potrebno je na donjem panelu

odabrati `Search` te umjesto `All` postaviti `HTTP Fuzz Results`. Za pretraživanje možemo na primjer unijeti `[1-9] records found` što će pronaći odgovor s više od 0 pronađenih stavki.



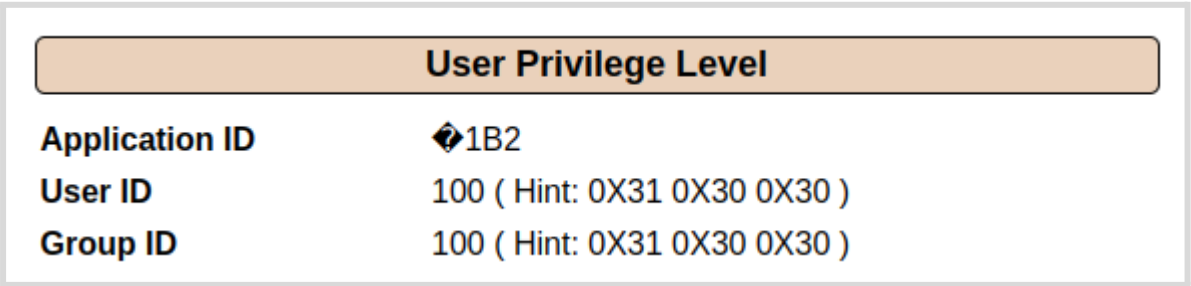
U našem slučaju to je 23 stavki. Dakle, aplikacija je ranjiva na MySQL Injection napad. Ovaj konkretni zahtjev možemo lako otvoriti u pregledniku tako da na njega desnim klikom odaberemo `Open URL in Browser`. Ukoliko nas zanimaju konkretni parametri kojim je izveden napad, potrebno je dvaput kliknuti na nađeni zahtjev te pri vrhu odabrati `Header: Table (adv)`.



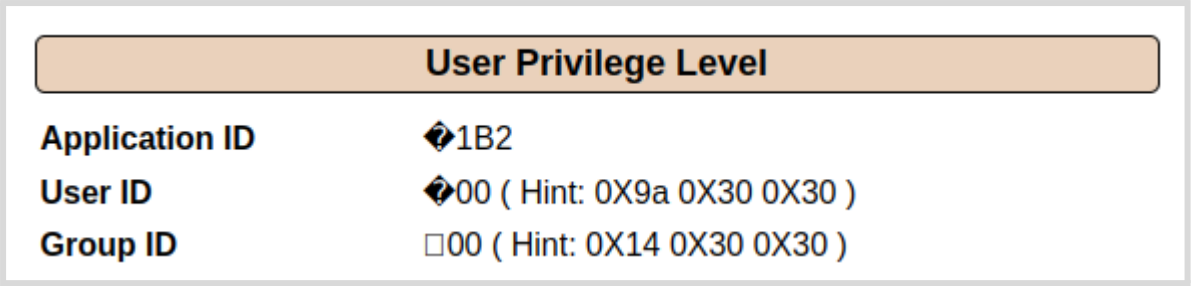
Napad možemo pronaći na CAPEC stranicama (https://capec.mitre.org/data/definitions/66.html). Kako bismo mitigirali ovaj napad, potrebno je validirati korisnički unos. Treba filtrirati navodnike i oznake komentara. Najbolje rješenje je koristiti parametrizirane upite s ograničenjima na parametre ili npr. `PreparedStatement` objekte u jeziku Java.

Parameter Addition / Tampering

Da bi napravili novo testiranje, vratit ćemo se na početnu stranicu (`Quick Start`), opet odabrati `Manual Explore` te pokrenuti Chrome. Sada ćemo odabrati OWASP 2017 > A1 - Injection (Other) > Parameter Addition > View User Privileges. Kao što i na stranici piše, cilj je izmijeniti zahtjev tako da User i Group ID budu `000` čime se eskaliraju privilegije. Primijetimo da u URL-u postoji parametar `iv` koji je postavljen na naizgled nasumičnu vrijednost. Probajmo promijeniti prva dva znaka, umjesto `6b` neka bude `aa`.

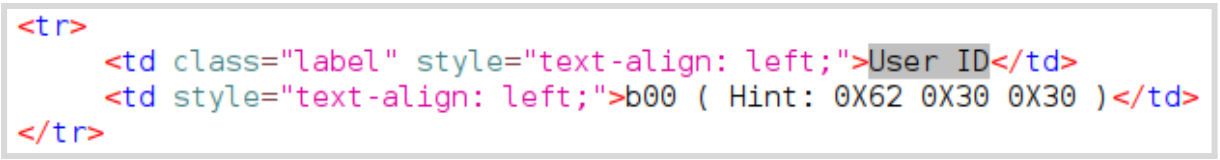


Promijenili smo prvi znak (tj. bajt) ID-a aplikacije. Na dobrom smo putu, međutim trebamo promijeniti prvi znak ID-a korisnika i grupe na 0. Postavimo sada 5. (`ab`) i 8. (`25`) *bajt* u parametru `iv` na `00`.



Pronašli smo bajtove koje trebamo testirati. Jedan bajt čine dvije heksadekadske znamenke. Zatvorimo sada prozor preglednika te pogledajmo povijest u ZAP-u, tj. zadnju stavku povijesti. U prikazanom zahtjevu označit ćemo prvu znamenku 5. bajta (`0`) te kliknuti `Fuzz…` kao i u prethodnom napadu. Za payload trebamo postaviti heksadekadske znamenke. Odaberimo File Fuzzers > jbrofuzz > Number Systems > Base16 (HEX). Isto ćemo napraviti za drugu znamenku.

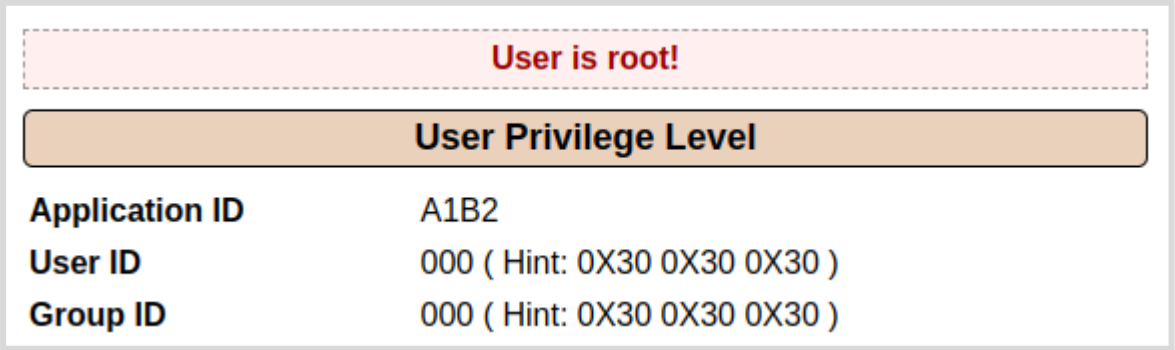
Imamo dakle, za sada, 2 lokacije, a svaka ima 16 mogućnosti. Pokrenimo neizrazito testiranje. Zatim moramo odlučiti što pretraživati kako bismo našli traženi rezultat. Možemo otvoriti bilo koji rezultat te potražiti `User ID` u odgovoru na zahtjev (CTRL + F).



Tražimo vrijednost `000` u drugom retku. Odaberimo sada pretraživanje HTTP Fuzz rezultata kao i u prethodnom napadu te potražimo: `<td style="text-align: left;">000`
Trebao bi se izlistati samo jedan zahtjev, kojeg za provjeru možemo otvoriti u pregledniku. Iz URL-a zahtjeva možemo zaključiti da 5. bajt treba biti `aa`.
Očistimo rezultate *fuzzera* klikom na ikonu metle na prikazu `Fuzzer`, te započnimo novi neizraziti test za 8. bajt. 5. bajt vratit ćemo na `00` kako bi nam bilo lakše pretraživati rezultate.

Rezultate pretražujemo s istim upitom kao i prije, te opet pronalazimo samo jedan rezultat. Sada možemo zaključiti da 8. bajt treba biti `24`.

Dakle, `iv` u URL-u valja postaviti na `6bc24fc1**aa**650b**24**b4114e93a98f1eba` kako bi napad bio izvršen.



Ova ranjivost prikazuje dva problema. Svaki korisnik koristi uvijek isti `iv` kako bi se autentificirao, a `iv` se bajt po bajt preslikava u ID korisnika. Ranjivost se može spriječiti drukčijim sustavom autentifikacije, pr. korištenjem *Session ID*-a. Prilikom uspješnog ulogiravanja poslužitelj stvara sesiju te vraća ID sesije korisniku. ID je nasumičan i nije koreliran s ID-om korisnika. Sesija je privremena, a nakon isteka je potrebno ponovno ulogirati se. *Session ID* može biti parametar u URL-u, ili bolje, pohranjen kao kolačić.

Reflected XSS

Pokrenimo `Manual Explore` pomoću preglednika Chrome još jednom. Ovaj put ćemo odabrati OWASP 2017 > A7 - Cross Site Scripting (XSS) > Reflected (First Order) > Set Background Color. Isprobajmo primjer, unesimo `FF0000` te kliknimo na gumb. Pozadina će postati crvena. Zatvorimo sada prozor te se vratimo u ZAP sučelje. U povijesti je lako pronaći naš POST zahtjev za crvenu pozadinu.

Ovaj put parametar nije dio URL-a, već je u *payloadu*. Označimo ga, te desnim klikom odaberimo `Fuzz…`. Za *payload* ćemo postaviti File Fuzzers > jbrofuzz > XSS > XSS 101. Sada možemo pokrenuti neizraziti test koji bi trebao završiti unutar nekoliko sekundi. Nakon završetka, odaberimo pretragu `HTTP Fuzz Results`, te potražimo: `<script>`. Na žalost, ZAP nam ne omogućava da pretražimo one rezultate koji vraćaju *alert*, čime bismo dokazali ovaj XSS napad. Stoga, najbolje što možemo jest pretražiti `<script>` te manualno provjeriti ima li *alerta*. U nastavku je primjer pronađenog odgovora koji u HTML-u sadrži traženu riječ:

```
      </tr>
      <tr><td> </td></tr>
      <tr>
           <td class="informative-message" colspan="2" style="text-align: center;">
                The current background color is '> <script>alert(3)</script>                </td>
      </tr>
      <tr><td> </td></tr>
</table>
```

Kopirat ćemo vrijednost parametra za ‘boju’ te otvoriti URL pomoću padajućeg izbornika. Na novootvorenoj stranici zalijepit ćemo ovu vrijednost te kliknuti na gumb.

Dokazali smo XSS ranjivost.

Ovaj napad također možemo pronaći na CAPEC stranicama (https://capec.mitre.org/data/definitions/591.html). Korisnički unos potrebno je provjeravati. Pr. u ovom slučaju za boju pozadine očekujemo 6 alfanumeričkih znakova prema tome `<` ne bismo trebali prihvaćati. Također, možemo iz korisničkog unosa izbaciti nizove koji sadrže skriptne oznake i Javascript funkcije. Za najveću sigurnost možemo koristiti tehnologije koje zabranjuju vršenje skripti na klijentskoj strani, međutim time gubimo na funkcionalnosti stranice.