Univerzitet u Beogradu

**Fakultet organizacionih nauka**

Laboratorija za multimedijalne komunikacije

Predmet: Zaštita računarskih sistema

Projektni zadatak

Tema: Ranjivosti koje postoje unutar web aplikacija



Mentor: Bojan Marčeta

Student: Dragoslav Janković 294/2016

Sadržaj

[1. Opis aplikacije 3](#_Toc78211332)

[2. Ranjivosti 3](#_Toc78211333)

[2.1. SQL Injection 3](#_Toc78211334)

[2.1.1. SQL Injection zasnovan na 1=1 Always True 4](#_Toc78211335)

[2.1.2. SQL Injection zasnovan na “”=”” Always True 4](#_Toc78211336)

[2.1.3. SQL Injection zasnovan na nadovezivanju upita 4](#_Toc78211337)

[2.2. Cross-Site Scripting (XSS) i HTML Injection 5](#_Toc78211338)

[2.3. Ranjivosti prilikom „upload“-ovanja fajlova 6](#_Toc78211339)

[2.4. Broken Authentication 7](#_Toc78211340)

[2.4.1. Credential Stuffing 7](#_Toc78211341)

[2.4.2. Session timeouts 7](#_Toc78211342)

[2.4.3. Password Leaks 7](#_Toc78211343)

[2.5. Broken Access Control 8](#_Toc78211344)

[3. Primena alata iz “Kali Linux”-a 9](#_Toc78211345)

[3.1. Brute Force napadi korišćenjem „Burp Suite“ alata 9](#_Toc78211346)

[3.2. Pronalaženje skrivenih fajlova korišćenjem OWASP ZAP-a 12](#_Toc78211347)

[3.3. Detekcija Firewall-a pomoću “WAFW00F” alata 14](#_Toc78211348)

[4. Literatura 15](#_Toc78211349)

# 1. Opis aplikacije

Aplikacija je pisana u programskom jeziku PHP. Unutar aplikacije korisnici se mogu registrovati i prijaviti na sajt korišćenjem svoje email adrese i izabrane šifre. Aplikacija predstavlja mini web prodavnicu gde korisnici mogu dodavati proizvode u korpu i na kraju izvršiti plaćanje svih proizvoda iz korpe putem PayPal-a.

Pored običnih korisnika postoje administratori koji upravljaju ponuđenim proizvodima unutar aplikacije.

# 2. Ranjivosti

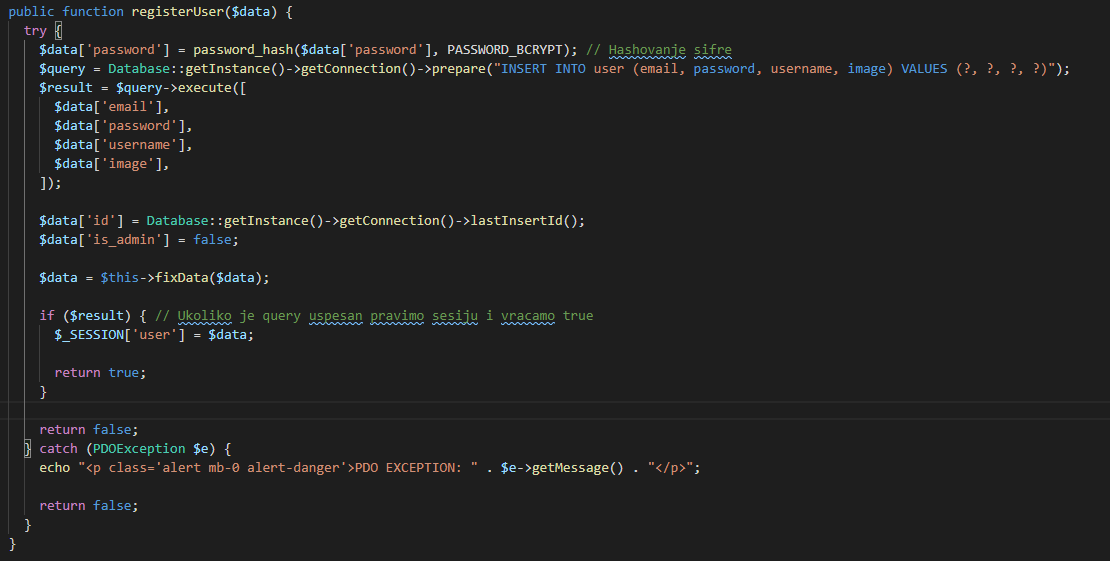
Već prilikom registracije nailazimo na nekoliko bitnih ranjivosti unutar web aplikacija na koje se mora obratiti pažnja. Prvo ćemo pogledati najpoznatije napade prema OWASP (Open Web Application Security Project) zajednici, a to su „INJECTIONS“.

## 2.1. SQL Injection

Najpopularnija vrsta „injectiona“ je „SQL Injection“. Postoji više načina na koji se može iskoristiti „SQL Injection“.

Aplikacija je zaštićena od „SQL Injection“ napada korišćenjem „Prepared Statement“-a i „PDO“-a.

Prikazaćemo primer kako je deo vezan za registraciju korisnika zaštićen:



Da naša baza nije zaštićena kao što je prikazano u primeru gore mogli bi se desiti neki od sledećih „SQL Injection“-a.

### 2.1.1. SQL Injection zasnovan na 1=1 Always True



Unošenjem stringa „gagi OR 1=1“ u polje za „username“ možemo izvršiti SQL upit koji će uvek biti izvršen uspešno i izgledaće ovako:

„SELECT \* FROM user WHERE username = gagi OR 1=1;“

Biće nam uspešno vraćeni svi redovi iz tabele user.

### 2.1.2. SQL Injection zasnovan na “”=”” Always True

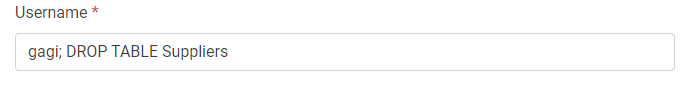


Unošenjem stringa „ “ OR ““=“ „ u polje za „username“ možemo izvršiti SQL upit koji će uvek biti izvršen uspešno i izgledaće ovako:

„SELECT \* FROM user WHERE username = "" OR ""="";“

Biće nam uspešno vraćeni svi redovi iz tabele user.

### 2.1.3. SQL Injection zasnovan na nadovezivanju upita



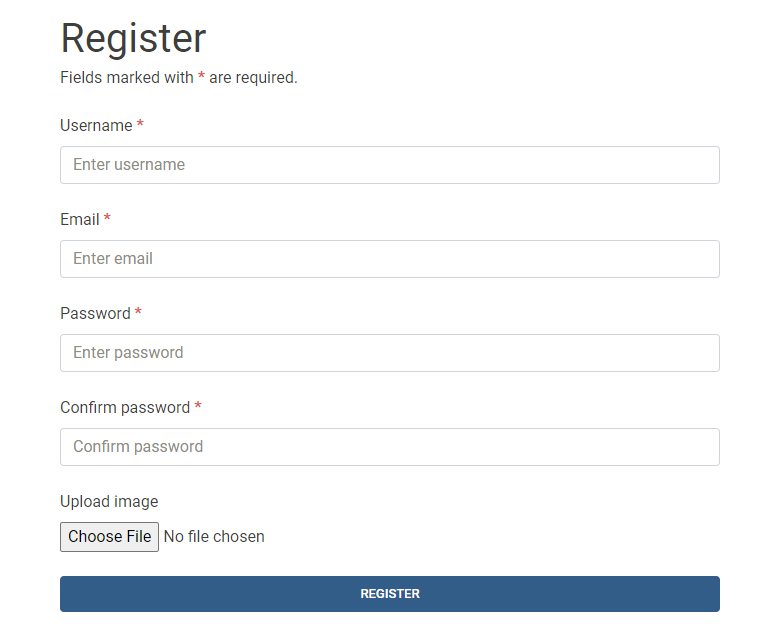
Unošenjem stringa „gagi; DROP TABLE Suppliers“ u polje za „username“ možemo izvršiti SQL upit koji će uvek biti izvršen uspešno i izgledaće ovako:

„SELECT \* FROM user WHERE username = gagi; DROP TABLE user;“

Tabela user će biti obrisana.

## 2.2. Cross-Site Scripting (XSS) i HTML Injection

XSS i HTML „injection“ napadi su jedni od najjednostavnijih napada koji postoje i oni se mogu veoma lako izbeći. Suština ovih napada je ubacivanje malicioznog HTML ili Javascript koda preko input polja koje se šalje na server. Ukoliko na serveru ne postoji odgovarajuća sanitizacija input polja, korisnik to može zloupotrebiti.



Na primer, korisnik može uneti u polje „username“ nešto poput **<script>alert(“Hello World.”)</script> ili <h1>Hello World.</h1>**.

Ukoliko registracija bude uspešna ovaj kod će se sačuvati u bazi i izvršiće se svaki put kada se negde u aplikaciji bude koristilo korisničko ime novoregistrovanog korisnika.

Ovaj problem se rešava pomoću predefinisane funkcije u PHP-u koja se zove „htmlspecialchars“



Kao parametar prima string vrednost i primljeni string sanira tako da sve specijalne znakove pretvori u HTML karakter kodove. Zbog toga HTML će uspeti da pročita ono što je napisano, ali se kod neće zapravo izvršiti.

Ovako izgleda korisničko ime u bazi:



&lt; je zamena za <

&gt; je zamena za >

A ovako to izgleda u samoj aplikaciji:

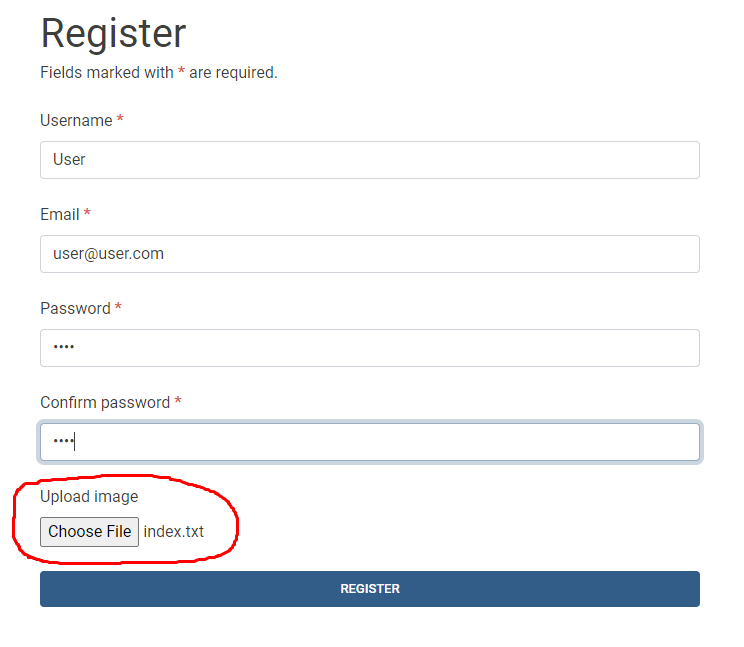


Prikazan je najprostiji slučaj u kome se izvršava samo „alert“ funkcija. Moguće je napisati mnogo kompleksnije Javascript injectione koji mogu uništiti celu aplikaciju.

## 2.3. Ranjivosti prilikom „upload“-ovanja fajlova

U delu za registraciju korisnika korisnicima je dozvoljeno da „upload“-uju svoju profilnu sliku. Svako dodavanje fajlova na server mora strogo da se iskontroliše. Potrebno je pre svega proveriti da li korisnik uopšte šalje sliku ili neki maliciozni fajl. To možemo proveriti preko ekstenzije fajla koji je poslat na server. Takođe, pored tipa fajla važno je proveriti i koliko je sam fajl velik. Treba odrediti maksimalnu dozvoljenu veličinu fajlova koju će server primati.

U ovom slučaju pokušaćemo da dodamo tekstualni fajl na server.

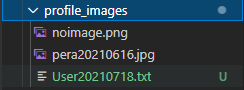


Pošto na serveru ne postoje nikakve provere prilikom dodavanja fajlova registracija će biti uspešno izvršena. Pošto smo u ovom primeru dodavali tekstualni fajlt posledice neće biti drastične, ali će svakako dovesti do problema u aplikaciji.

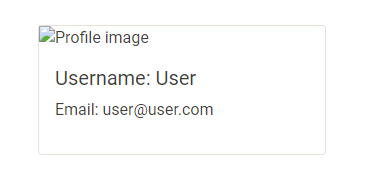
U bazi će to izgledati ovako:



Na serveru će biti dodat tekstualni fajl:



Problem će nastati kada u aplikaciji bude potrebno da se učita slika koji je korisnik poslao i to će izgledati ovako:



Druga opcija je bila dodavanje prevelikog fajla što može dovesti do rušenja celog servera i problema sa bazom.

## 2.4. Broken Authentication

Ova vrsta napada omogućava korisnicima da zaobiđu proces autentifikacije i da se uspešno prijave na neki od postojećih naloga i izvršavaju razne malverizacije.

Navešćemo nekoliko primera “Broken Authentication”-a.

### 2.4.1. Credential Stuffing

Napadač pokušava da pristupi aplikaciji koristeći korisnička imena i šifre sa liste najpopularnijih i najslabijih kombinacija. Zbog toga je bitno korisnicima prilikom registracije naglasiti da izaberu dovoljno jaku i jedinstvenu lozinku za svoj nalog. Ukoliko korisnik koristi šifru “123456” postoje velike šanse da mu nalog bude hakovan i zloupotrebljen.

### 2.4.2. Session timeouts

Ukoliko ne postoji period nakon koga sesija prestaje da bude važeća mogu se javiti potencijalni problemi u aplikaciji. Korisnik može pristupiti aplikaciji sa računara koji se nalazi na fakultetu, biblioteci itd. i zaboraviti da se odjavi. Pošto ne postoji vreme nakon koga će sesija prestati da važi sledeća osoba koja sedne za isti računar će imati pristup nalogu prethodnog korisnika.

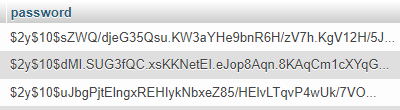
### 2.4.3. Password Leaks

Ukoliko napadač uspe da dobije pristup bazi podataka potrebno je da svi osetljivi podaci budu “hash”-ovani. Ukoliko se u bazi šifra čuva u originalnom obliku koliko god kompleksa ona bila biće vidljiva napadaču koji je dobio pristup bazi. Zbog toga se šifre “hash”-uju.

Naša aplikacija koristi “BCRYPT” algoritam za “hash”-ovanje šifri.



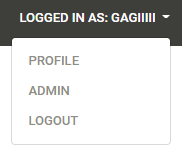
U bazi su šifre sačuvane u sledećem obliku:



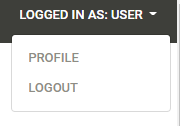
## 2.5. Broken Access Control

Jedna od ranjivosti aplikacija je loše implementirana kontrola pristupa raznim delovima aplikacije. Delovi aplikacije koji nisu namenjeni običnim korisnicima moraju biti zaštićeni, takođe svaki korisnik ima listu stvari koje sme da menja i listu stvari kojima mu je zabranjen pristup.

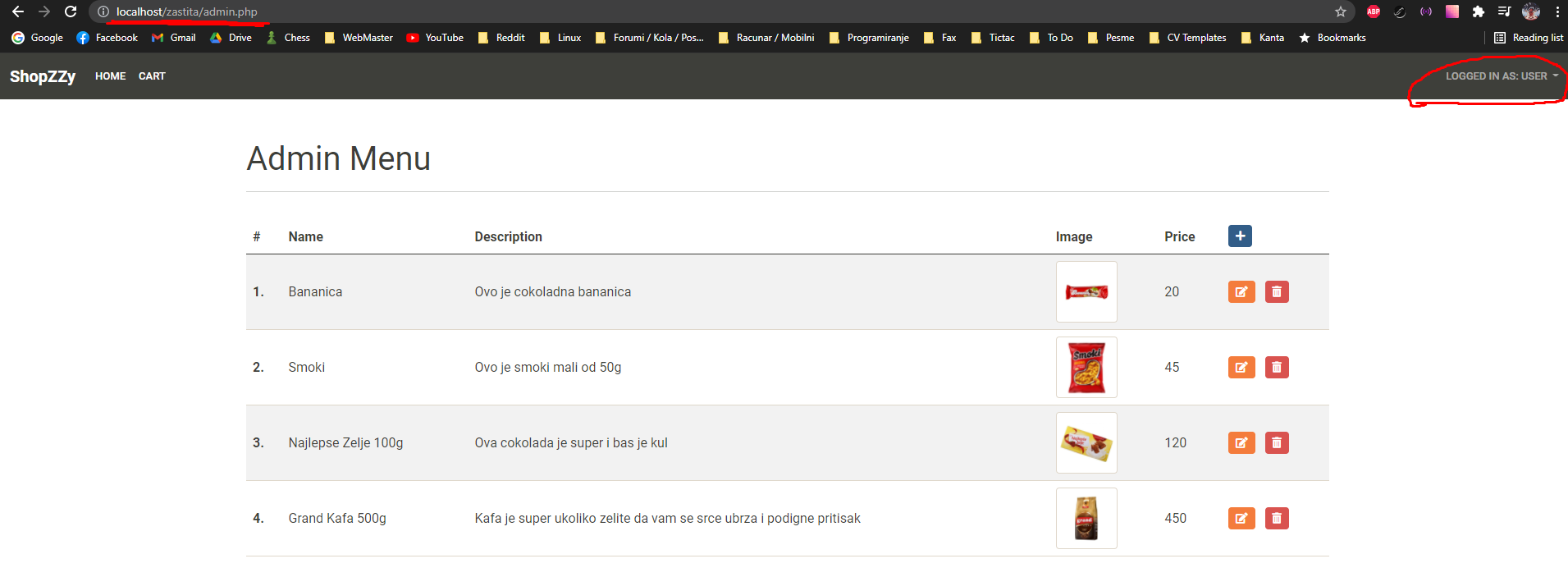
U našem primeru prikazaćemo kako običan korisnik može da pristupi admin delu što nikako ne bi smelo da se dogodi.



Korisnik „GaGiiiii“ je administrator i u svom meniju ima opciju za admin deo.



Korisnik „User“ nije administrator i u svom meniju nema opciju za admin deo. Međutim iako nismo ulogovani preko admin naloga svakako možemo da pristupimo admin delu. Izmenom URL-a lako pristupamo admin delu jer u kodu za admin skriptu ne postoji nikakav vid kontrole pristupa.



Ovo se može sprečiti jednostavnom proverom pre početka izvršavanja skripte.

$loggedIn = Database::getInstance()->isUserLoggedIn();

if (!$loggedIn || ($loggedIn && !$\_SESSION['user']['user.is\_admin'])) {

header("Location: index.php);

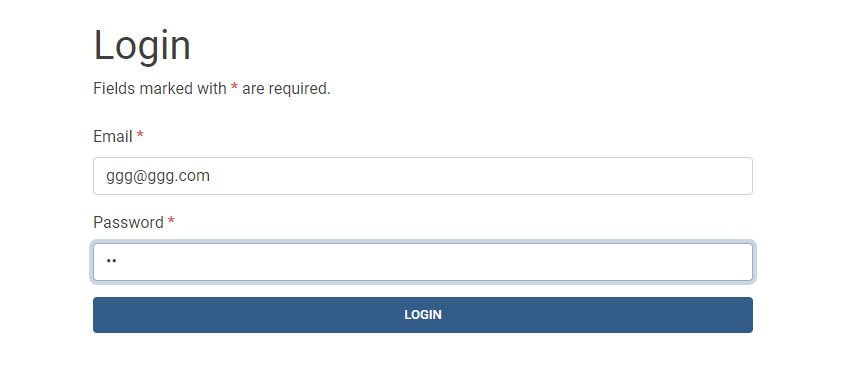
exit;

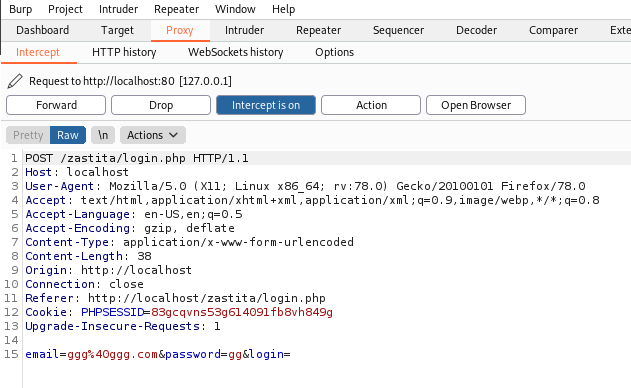
}

# 3. Primena alata iz “Kali Linux”-a

## 3.1. Brute Force napadi korišćenjem „Burp Suite“ alata

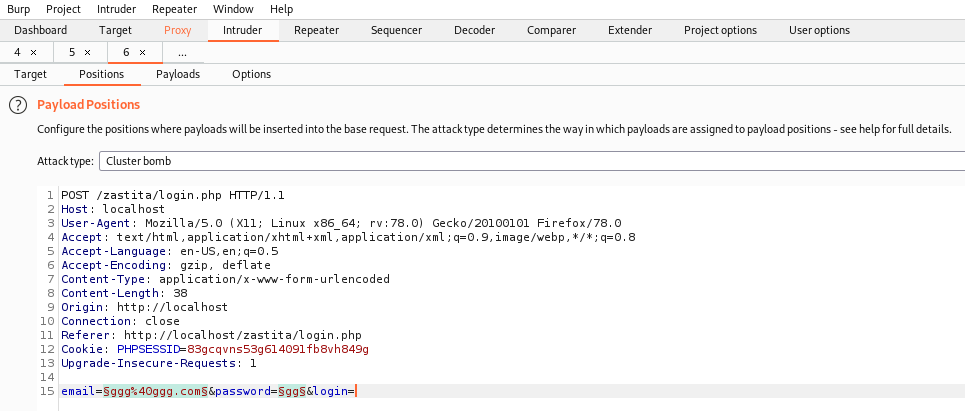
Unutar Burp Suite-a ćemo otići u Proxy tab i kliknuti dugme Intercept on, zatim ćemo na našem sajtu na login stranici ukucati podatke za login (nebitno da li su ispravni podaci i stvarni).



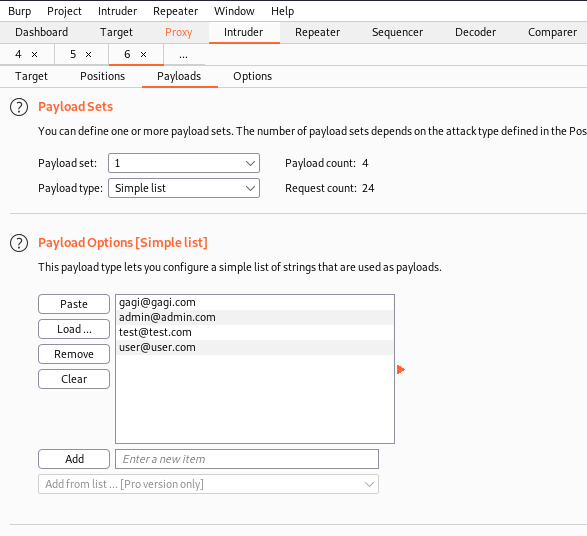


Sledeći korak je desni klik bilo gde unutar prozora i izbor stavke „send to intruder“.

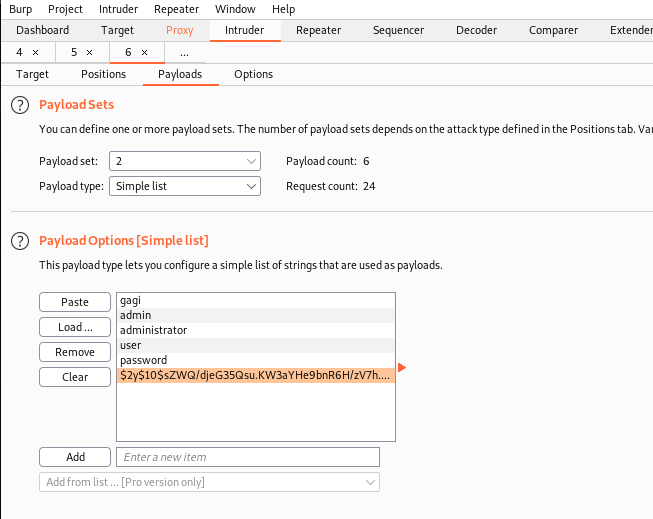
Burp Suite automatski selektuje polja koja treba da se brute force-uju, zbog toga ćemo prvo clear-ovati sva polja i zatim ručno selektovati polja za email i šifru i kliknuti na dugme add. Nakon selektovanja polja ona bi trebalo da pozelene. Izabraćemo Cluster Bomb vrstu napada.



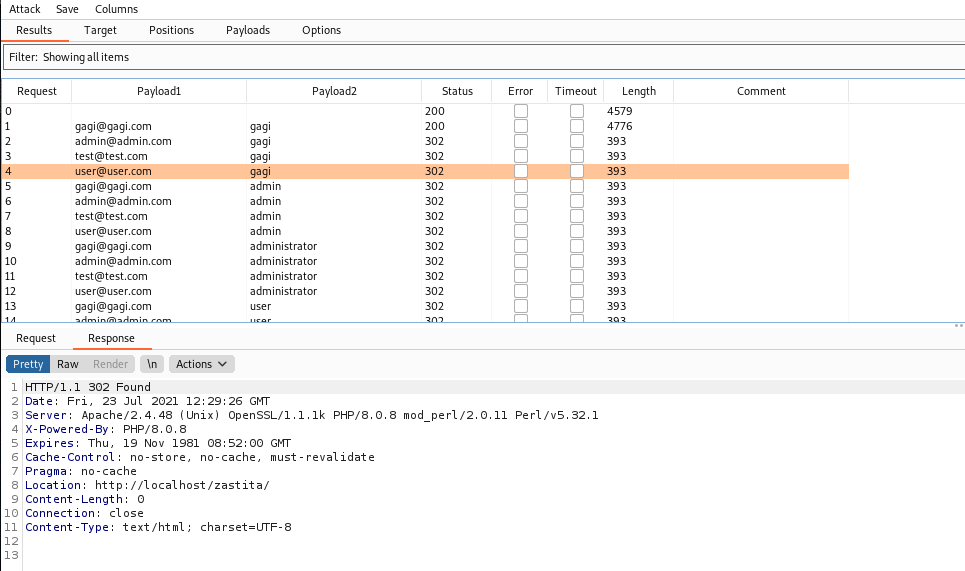
Preći ćemo na tab payloads i izabrati payload set 1 koji se odnosi na email. U payload options možemo dodati listu emailova koje želimo da koristimo prilikom napada. Postoje već napravljene liste koje sadrže najpopularnije emailove, ali mi ćemo koristiti samo par emailova koje ćemo ručno dodati.



Sada ćemo izabrati payload set 2 i isti postupak ponoviti za šifre koje želimo da koristimo prilikom napada.

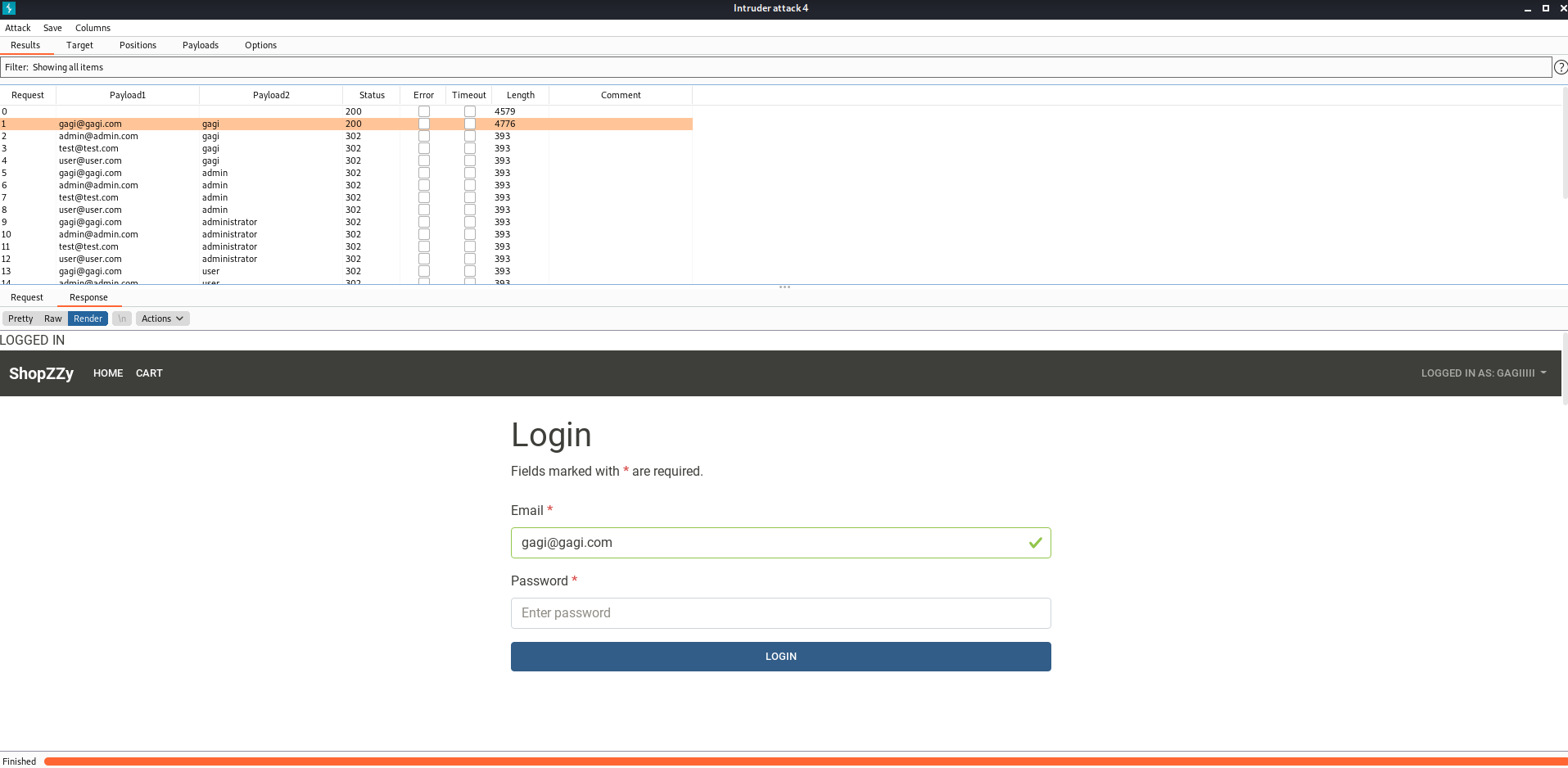


Sada ćemo kliknuti dugme start attack i dobićemo sledeći odgovor.



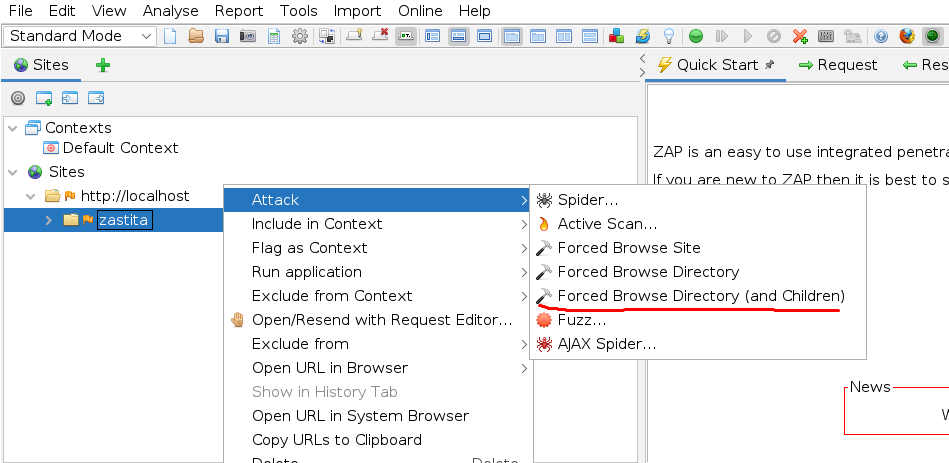
Vidimo da se prvi red razlikuje od ostalih. Ima status 200 i drugačiju dužinu od ostalih.

Kada kliknemo na njega videćemo sve informacije o zahtevu. Kada kliknemo na response tab videćemo i i kako u stvarnosti izgleda odgovor od servera, koji će nam ispisati poruku logged in.

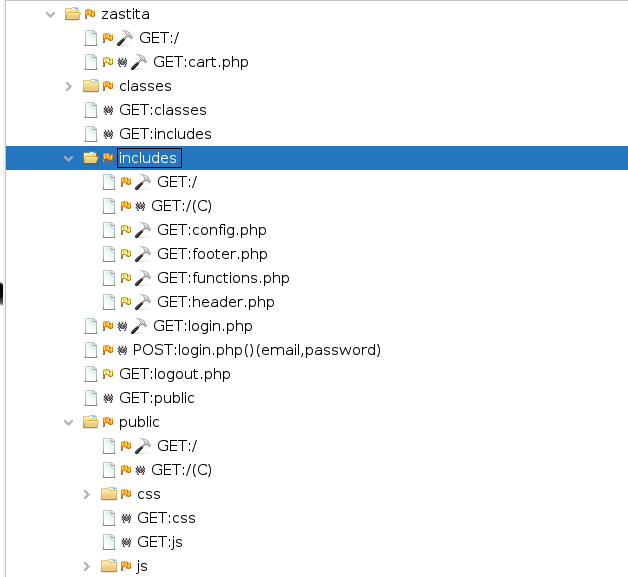
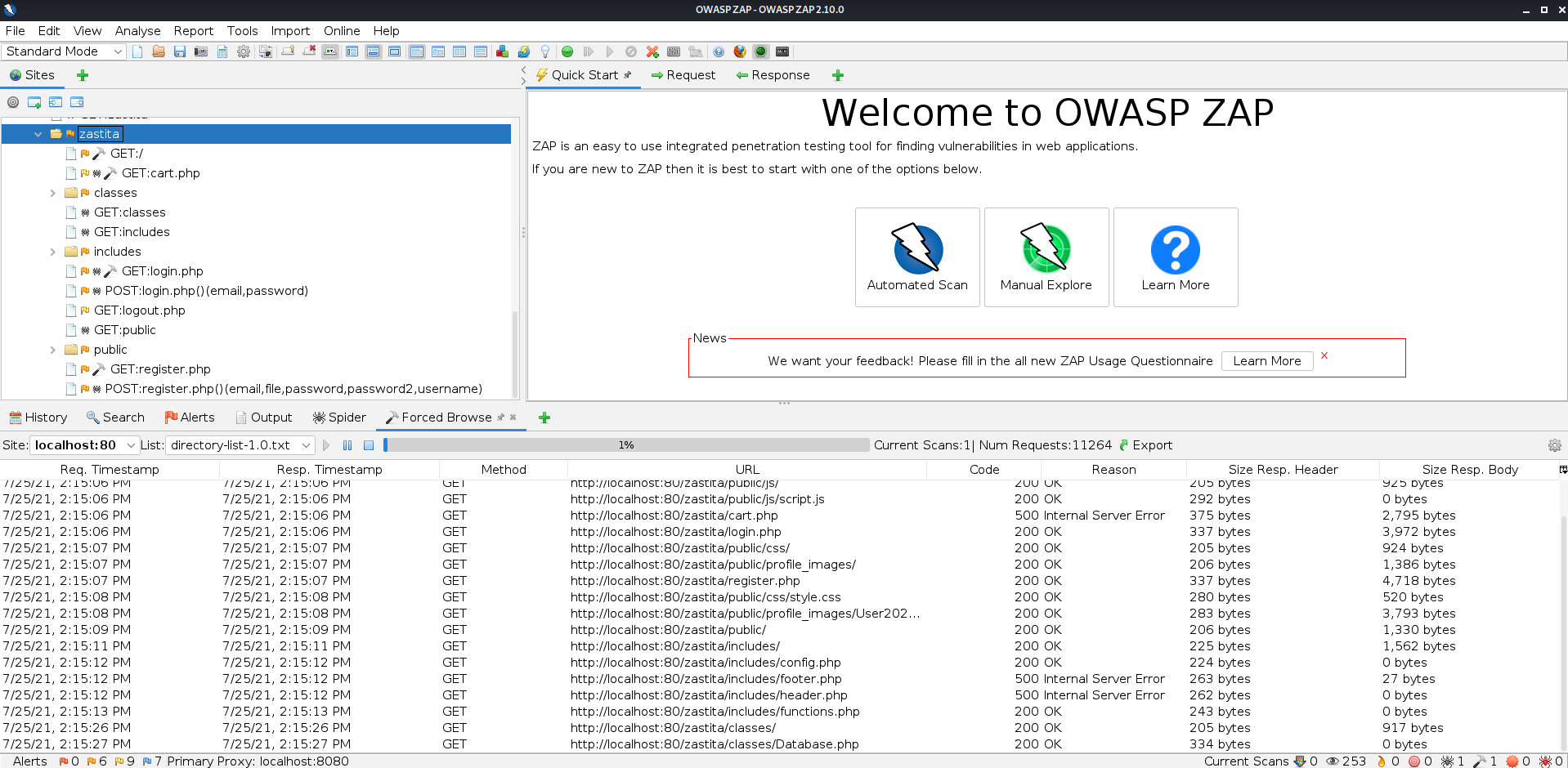


## 3.2. Pronalaženje skrivenih fajlova korišćenjem OWASP ZAP-a

Pokrenućemo ZAP program, nakon podešavanja proxy-ja biće nam prikazan localhost folder i naša aplikacija unutar njega. Desnim klikom na folder naše aplikacije biramo „Forced Browse Directory (and Children).



Nakon uspešno završenog skeniranja dobićemo sledeći rezultat:



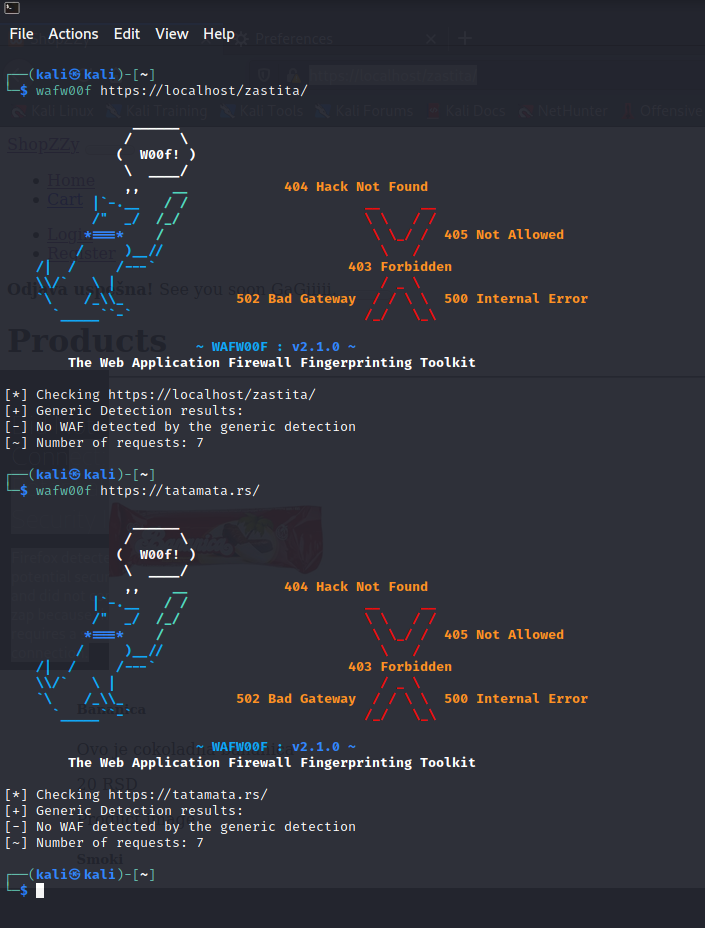
Kao što vidimo pronađeni su svi fajlovi unutar naše aplikacije. Pronađeni su čak i server fajlovi i konfiguracioni fajlovi. To može da predstavlja problem ukoliko pristup njima nije strogo kontrolisan.

Na primer na serveru može postojati fajl “createDatabase.php” koji svaki put pravi novu bazu.

Pronalaženje takvog ili sličnog fajla i pristup njemu mogu napraviti veliki problem aplikaciji.

## 3.3. Detekcija Firewall-a pomoću “WAFW00F” alata

Detekcija Firewall-a se vrši veoma lako pomoću navedenog alata. Potrebno je samo da pokrenemo terminal i ukucamo “wafw00f URL” i alat će izvršiti skeniranje sajta i poslati nam odgovor.



Kao što vidimo izvšene su provere na 2 sajta. Prvi se nalazi u našem localhost-u, a drugi je realan sajt hostovan na serveru. U oba slučaja nije detektovan firewall.

# 4. Literatura

1. Stack Overflow forum, <https://stackoverflow.com/>
2. Cheatsheet web site <https://gagiiiii.github.io/cheatsheet/>
3. OWASP Top Ten <https://owasp.org/www-project-top-ten/>
4. Kali Linux <https://www.kali.org/>