Postupci zaštite web aplikacije

Glavaš, Vedran

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

POSTUPCI ZAŠTITE WEB APLIKACIJE

Diplomski rad

Vedran Glavaš

Osijek, 2018.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 25.03.2018.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Vedran Glavaš
Studij, smjer:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. studenta, godina upisa:	D 991, 20.09.2017.
OIB studenta:	73580003825
Mentor:	Izv. prof. dr. sc. Krešimir Nenadić
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc.dr.sc. Alfonzo Baumgartner
Član Povjerenstva:	Doc.dr.sc. Ivica Lukić
Naslov diplomskog rada:	Postupci zaštite web aplikacije
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak diplomskog rada:	Navesti pregled postupaka zaštite web aplikacije. Za svaki postupak navesti primjer, prednosti i nedostatke te moguća alternativna rješenja. Navesti preporuke prosječnom web korisniku i programeru za nekoliko razina zaštite web aplikacije (minimalna, preporučena, visoka razina zaštite).
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Vrlo dobar (4)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 1 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 1 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	25.03.2018.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 16.04.2018.

Ime i prezime studenta:	Vedran Glavaš
Studij:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. studenta, godina upisa:	D 991, 20.09.2017.
Ephorus podudaranje [%]:	25%

Ovom izjavom izjavljujem da je rad pod nazivom**: Postupci zaštite web aplikacije**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Krešimir Nenadić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Web aplikacije predstavljaju programska rješenja kojima se pristupa putem internet preglednika koristeći internet. Činjenica da su dostupne u bilo koje vrijeme s bilo kojeg mjesta, računala i mobilnih telefona dovela je do ubrzanog razvoja i rasta *web* aplikacija. Njihova popularnost i široka primjena očituje se kroz internet bankarstvo, internet trgovine, društvene mreže, udaljeni pristup podacima i sl. Korištenje takvih *web* aplikacija često podrazumijeva i razmjenu povjerljivih informacija što može predstavljati veliki sigurnosni problem.

U ovom su radu objašnjene neke od metoda napada na *web* aplikacije te načini kako se zaštititi od mogućih napada.

Rad se sastoji od šest poglavlja od kojih je prvo poglavlje uvodno. U drugom je poglavlju definiran pojam sigurnosti i opisani su osnovni elementi sigurnosti. U trećem poglavlju opisane su metode napada na *web* aplikaciju, dok su u četvrtom poglavlju opisani koraci koje je moguće poduzeti kako bi se povećala sigurnost aplikacije. U petom su poglavlju opisani neki od besplatnih alata koji se mogu koristiti za testiranje sigurnosti *web* aplikacije. Zaključak rada dan je u posljednjem, šestom, poglavlju.

1.1. Zadatak diplomskog rada

Navesti pregled postupaka zaštite *web* aplikacije. Za svaki postupak navesti primjer, prednosti i nedostatke te moguća alternativna rješenja. Navesti preporuke prosječnom *web* korisniku i programeru za nekoliko razina zaštite *web* aplikacije (minimalna, preporučena, visoka razina zaštite).

2. OSNOVE SIGURNOSTI WEB APLIKACIJA

Sigurnost je stupanj zaštite imovine od opasnosti, štete, gubitka ili kriminalne aktivnosti. Vezana je uz upravljanje rizicima i primjenu učinkovitih protumjera.

Osnovni elementi sigurnosti su [1]:

- **Autentifikacija**
  - Autentifikacija je proces utvrđivanja identiteta korisnika *web* aplikacije. Osigurava mogućnost da se nepobitno utvrdi da je sudionik u komunikaciji doista onaj za kojega se predstavlja.
- **Autorizacija**
  - Autorizacija je proces koji upravlja resursima i operacijama kojima smiju pristupiti samo ovjereni korisnici. Resursi uključuju datoteke, baze podataka, tablice, konfiguracijske podatke, itd. Operacije uključuju obavljanje transakcija kao što je kupnja proizvoda, prijenos novca s jednog računa na drugi, itd.
- **Provjera**
  - Učinkovita provjera i zapisivanje ključ su za jednoznačno povezivanje svih operacija nekog korisnika s tim korisnikom tj. korisnik nakon što izvrši određenu operaciju nije u mogućnosti opovrgnuti da je tu operaciju izvršio.
- **Povjerljivost**
  - Povjerljivost osigurava da podaci ostaju privatni i povjerljivi te da im ne mogu pristupiti neovlašteni korisnici. Šifriranje podataka se često koristi za provođenje povjerljivosti.
- **Integritet**
  - Integritet je jamstvo da su podaci zaštićeni od slučajnih ili namjernih (zlonamjernih) promjena.
- **Dostupnost**
  - U pogledu sigurnosti, dostupnost znači da sustavi ostaju dostupni za legitimne korisnike. Mnogi napadači za cilj imaju uskratiti usluge aplikacije legitimnim korisnicima.

3. RANJIVOSTI WEB APLIKACIJA

3.1. Napad umetanjem SQL koda

Napad umetanjem SQL koda (eng. *SQL Injection*) predstavlja veliku prijetnju za *web* stranice koje u svojem radu koriste SQL (eng. *Structured Query Language*) baze podataka. Uspješnim izvršavanjem ovog napada moguće je doći do povjerljivih informacija, modificirati podatke u bazi ili izvršavati operacije nad bazom podataka poput brisanja tablica ili brisanja same baze.

Napad utječe na [2]:

- **Povjerljivost** – u bazama podataka se najčešće nalaze povjerljive informacije poput osobnih podataka o korisnicima. Napadači mogu otkriti informacije o adresama elektroničke pošte korisnika što se kasnije može iskoristiti za dodatne napade.
- **Autentifikaciju** – napad umetanjem SQL koda se koristi kako bi se zaobišla prijava u sustav. Napadač se tako može prijaviti kao neki drugi korisnik bez poznavanja odgovarajuće zaporke.
- **Autorizaciju** – napadač može iskoristiti podatke u bazi kako bi se prijavio kao administrator i tako dobio dodatne ovlasti nad bazom ili *web* stranicom.
- **Integritet podataka** – jednako kao što napadač može čitati podatke iz baze, može ih i mijenjati. U krajnjem slučaju, napadač može uništiti cijelu bazu podataka.

Najčešći oblici napada umetanjem SQL koda su [2]:

- Umetanje SQL koda u polje za unos podataka
- Umetanje SQL koda u URL (eng. *Uniform Resource Locator*) adresu

3.1.1. Umetanje SQL koda u polje za unos podataka

Najpoznatiji oblik napada umetanjem SQL koda je umetanje pomoću polja za unos podataka. Ukoliko *web* stranica ne koristi metode za provjeru i validaciju unesenih podataka, napad je jednostavniji za izvesti.

Napadač se, korištenjem ovog oblika napada, može prijaviti u sustav bez poznavanja odgovarajućeg korisničkog imena ili zaporke.

Neka se na *web* stranici nalazi sljedeći HTML (eng. *Hypertext Markup Language*) kod za prijavu u sustav u kojem se od korisnika zahtjeva unos korisničkog imena i zaporke:

```
<!-- HTML FORMA ZA PRIJAVU U SUSTAV -->

<form method="POST" action="login.php" name="login_form">

    <!-- POLJE ZA UNOS KORISNIČKOG IMENA -->

    <input type="text" name="ime">

    <!-- POLJE ZA UNOS ZAPORKE -->

    <input type="password" name="zaporka">

    <!-- GUMB ZA PRIJAVU U SUSTAV -->

    <input type="submit" name="btn_submit">

</form>
```

Pritiskom na gumb, podaci se šalju stranici *login.php* koja sadrži sljedeći kod za obradu zahtjeva:

```
<?php

$name = $_POST['ime'];

$password = $_POST['zaporka'];

$query = "SELECT name FROM korisnici WHERE name = '$name' AND password = '$password'";

?>
```

Kada se legitimni korisnik pokuša prijaviti u sustav, on će, primjerice, kao korisničko ime i zaporku upisati *Vedran* i *zaporka123*. Nakon slanja unesenih podataka, SQL upit će biti:

```
SELECT name  FROM korisnici WHERE name = 'Vedran' AND password = 'zaporka123'
```

Izvršavanjem ovog SQL upita pretražuje se tablica *korisnici* u bazi podataka sa vrijednostima atributa (*name* i *password*) *Vedran* i *zaporka123*. Ukoliko se na stranici ne vrši filtriranje i provjera

unesenih podataka, tada je moguće iskoristiti polja za unos podataka za izvršavanje bilo kakve SQL naredbe.

Ukoliko napadač u polja za unos podataka unese *' OR '1' = '1* SQL upit će biti sljedećeg oblika:

SELECT name FROM korisnici WHERE name = '' OR '1' = '1' AND password = '' OR '1' = '1'

Izvršavanjem ovakvog SQL upita baza neće uspoređivati podatke koje je korisnik unio, nego će provjeriti istinitost tvrdnje *'1' = '1'*. Kako je ta tvrdnja uvijek istinita, baza će kao odgovor vratiti prvi red u tablici *korisnici*, čime će se napadač uspješno prijaviti u sustav kao korisnik koji je prvi naveden u tablici. Cilj ovakvog napada je kreirati SQL upit koji će uvijek vraćati istinu u WHERE dijelu upita [2].

3.1.2. Umetanje SQL koda u URL adresu

Prilikom slanja zahtjeva prema bazi podataka *web* stranica kreira URL adresu određenog oblika u koju napadač može umetnuti zlonamjerni SQL kod.

Neka, na primjer, *web* stranica trgovine sportske opreme za prikaz određenog proizvoda kreira URL adresu oblika:

https://www.trgovina.hr/proizvodi/obuca?id=1

U ovom slučaju će stranica prikazati podatke o prvom proizvodu. Za prikaz informacija o nekom drugom proizvodu korisnik može u URL adresu umjesto *id=1* upisati, na primjer, *id=7* što će rezultirati prikazom informacija o sedmom proizvodu.

Upravo ova mogućnost mijenjanja URL adrese omogućuje napadačima umetanje zlonamjernih SQL kodova.

Za dohvaćanje podataka o proizvodu koristi se sljedeći SQL upit:

*"SELECT * FROM obuca WHERE id = ' " + product_id + " ' ";*

Umetanjem zlonamjernog SQL koda napadač može stvoriti SQL upit koji će rezultirati, na primjer, brisanjem tablice *obuca*. Kako bi to postigao, napadač može upisati sljedeći SQL kod:

3; DROP TABLE obuca

Umetanjem ovakvog SQL koda, URL adresa postaje:

https://www.trgovina.hr/proizvodi/obuca?id=3;%20DROP%20TABLE%20obuca

SQL upit prema bazi, nakon umetanja prethodnog SQL koda, ima sljedeći oblik:

*"SELECT * FROM obuca WHERE id = '3'; DROP TABLE obuca"*

Novi SQL upit sastoji se od dvije naredbe. Prva je naredba *SELECT* koja se koristi za dohvat podataka, a druga naredba je *DROP TABLE* koja će obrisati tablicu *obuca* i sve podatke koji se u njoj nalaze [2].

3.1.3. Primjeri napada umetanjem SQL koda

17. kolovoza 2009. Ministarstvo pravosuđa Sjedinjenih Američkih Država osudilo je američkog državljanina Alberta Gonzaleza i dva neimenovana državljana Rusije za krađu 130 milijuna brojeva kreditnih kartica pomoću napada umetanjem SQL koda. Napad, kojim su zahvaćeni i trgovački lanci trgovina 7-Eleven i Hannaford Brothers, opisan je kao najveći napad krađe identiteta u povijesti Sjedinjenih Američkih Država [3].

U razdoblju od 24. do 26. srpnja 2010. japanski i kineski napadači iskoristili su napad umetanjem SQL koda kako bi dobili pristup informacijama kreditnih kartica korisnika internetske trgovine Neo Beat. Napadom je zahvaćeno i sedam poslovnih partnera internetske trgovine. Krađa podataka zahvatila je 12191 korisnika usluga Neo Beat internetske trgovine [3].

28. lipnja 2013. turska hakerska grupa Redhack objavila je na svojem Twitter profilu kako su uspješno napali *web* stranicu uprave Istanbula. Tvrdili su da su uspjeli izbrisati dugove građana za vodu, plin, internet, električnu energiju i telefonske usluge. Osim toga, objavili su korisnička imena i zaporke administratora stranice kako bi se građani mogli prijaviti i izbrisati svoje dugove [3].

3.2. Napad umetanjem skriptnog koda

Napad umetanjem skriptnog koda (eng. *Cross-Site Scripting*), ili skraćeno XSS napad, jedan je od najčešćih napada na internetu. XSS napadom napadač u korisničkom *web* pregledniku izvršava

umetnuti programski kod, što mu omogućuje prikupljanje različitih osjetljivih podataka koji su dostupni pregledniku [4]. Umetnuti programski kod može biti JavaScript, HTML ili bilo koji drugi kod kojega *web* preglednik može izvršiti.

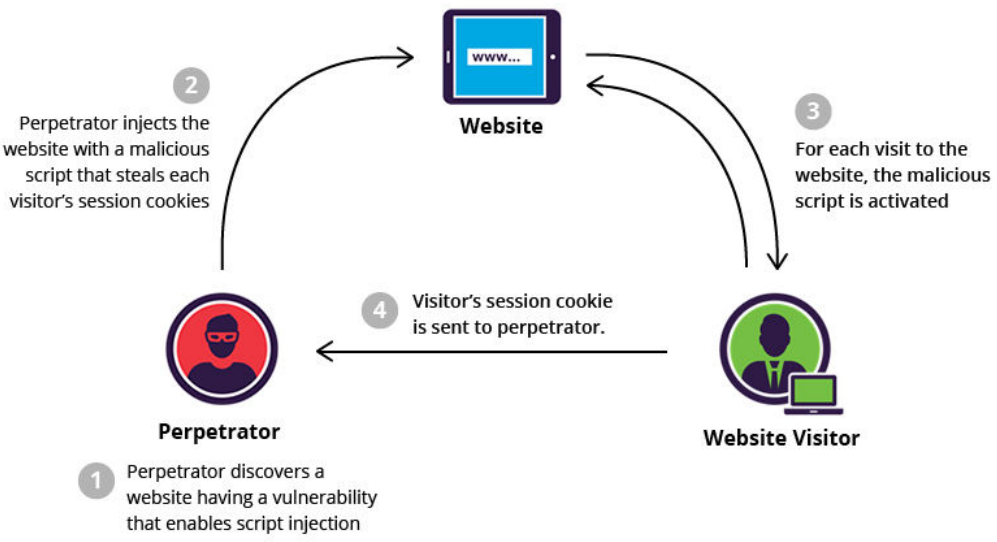
Zloupotreba XSS sigurnosnog nedostatka, prema [5], rezultira kompromitiranjem sigurnosne politike skriptnih jezika čiji se kod izvršava na strani klijenta tako da se korisnik ili njegovi povjerljivi podaci preusmjere s legitimnog na neki maliciozni *web* poslužitelj, čime se zaobilaze domenske restrikcije poslužitelja.

Dva su najpoznatija tipa XSS napada: trajni XSS napad (eng. *Persistent*) i jednokratni XSS napad (eng. *Non-Persistent*). Postoji i treći, manje poznati tip XSS napada: DOM (eng. *Document Object Model*) bazirani ili lokalni XSS napad [5].

3.2.1. Trajni XSS napad

Trajni XSS napad najopasniji je tip XSS napada. Nastaje kada napadač, iskorištavajući poznate ranjivosti *web* aplikacije, trajno spremi zlonamjerni kod na *web* poslužitelju. Najčešće mete napada su *web* stranice koje omogućuju korisnicima dijeljenje sadržaja, kao što su društvene mreže i mreže za dijeljenje videozapisa. Štetni se kod šalje korisniku kao rezultat njegovog HTTP (eng. *Hypertext Transfer Protocol*) zahtjeva [6].

Na slici 3.1. prikazan je primjer trajnog XSS napada.



Sl. 3.1. *Primjer trajnog XSS napada [6]*

Prema slici 3.1. napad se odvija kroz četiri faze:

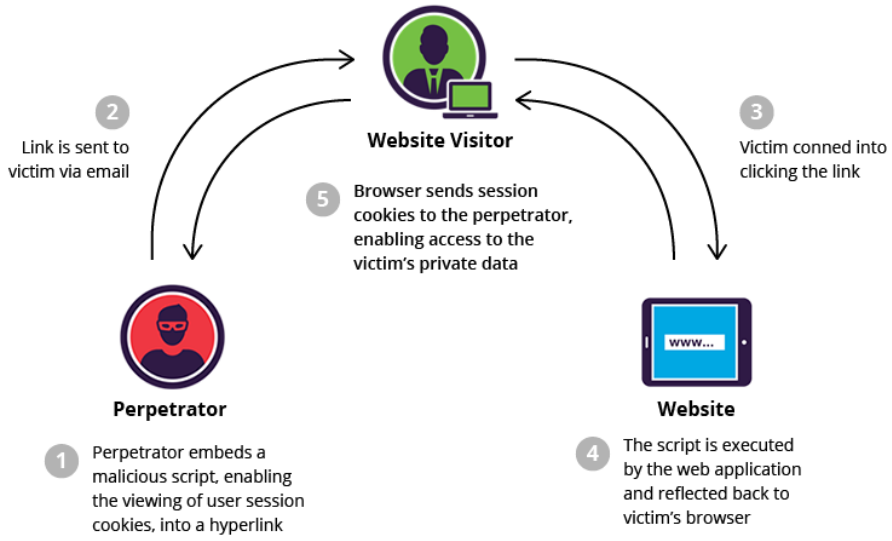
1. Napadač otkriva propust u *web* stranici koji mu omogućuje umetanje zlonamjernog koda
2. Napadač na *web* stranicu umeće zlonamjerni kod
3. Zlonamjerni kod se aktivira prilikom svakog posjeta stranici
4. Korisnički podaci se šalju napadaču

3.2.2. Jednokratni XSS napad

Najčešći oblik XSS napada je jednokratni XSS napad. Od trajnog se XSS napada razlikuje u tome što zlonamjerni skriptni kod nije trajno pohranjen na *web* poslužitelju.

Zlonamjerni kod se aktivira klikom na link koji šalje zahtjev stranici na kojoj je omogućeno izvršavanje koda. Maliciozni link može se distribuirati umetanjem u elektroničku poštu ili kao komentar na nekakvu objavu na društvenoj mreži [7].

Na slici 3.2. prikazan je primjer jednokratnog XSS napada.



Sl. 3.2. *Primjer jednokratnog XSS napada [7]*

Prema slici 3.2. jednokratni XSS napad se odvija kroz pet faza:

1. Napadač umeće zlonamjerni kod u link

2. Napadač šalje link potencijalnoj žrtvi u obliku elektroničke pošte
3. Žrtva klikom na link aktivira zlonamjerni kod
4. Zlonamjerni kod se izvršava u *web* aplikaciji
5. Žrtvin preglednik šalje povjerljive podatke napadaču

3.2.3. DOM bazirani XSS napad

DOM bazirani ili lokalni XSS napad predstavlja vrstu napada u kojoj se zlonamjerni kod ne izvršava na *web* stranici već lokalno na žrtvinom računalu.

Prilikom izvršavanja JavaScript koda *web* preglednik kreira objekte koji zajedno tvore DOM. Jedan od tih objekata je i objekt *document* pomoću kojeg je moguće pristupati postavkama *web* stranice. Neki od atributa koji su sadržani u *document* objektu, a mogu se iskoristiti za XSS napad, su atributi *location, URL* i *referrer* [5].

Prema [5], DOM bazirani XSS napad može se izvršiti ukoliko na *web* stranici postoji ugniježđeni JavaScript kod koji dohvaća vrijednost *document.URL* parametra. Ukoliko dohvaćena vrijednost parametra sadrži zlonamjerni JavaScript kod, a ne kodira se korištenjem odgovarajućih HTML entiteta prije ispisa na stranicu, taj upravo generirani HTML sadržaj bit će reinterpretiran od strane preglednika kao skriptni kod, a ne kao običan tekst, pa će se umetnuti zlonamjerni JavaScript kod zaista i izvršiti.

3.2.4. Primjer napada umetanjem skriptnog koda

Najpoznatiji primjer XSS napada je tzv. Samy crv kojega je razvio Samy Kamkar. Dizajniran je za propagaciju društvenom mrežom MySpace. Napad je pokrenut 4. listopada 2005. te je u manje od 20 sati zaraženo preko milijun korisnika društvene mreže. Crv je bio relativno bezazlen, a prikazivao je poruku "ali najviše od svega, Samy je moj junak" (eng. *But most of all, Samy is my hero*) na stranici profila žrtve. Uz prikaz poruke, crv je Samyju slao zahtjev za prijateljstvo. Kada je korisnik pogledao stranicu Samyjevog profila, crv se replicirao na njihov profil i tako se nastavio distribuirati mrežom [8].

3.3. Napad krivotvorenjem zahtjeva

CSRF (eng. *Cross-Site Request Forgery*) je vrsta napada koja iskorištava povjerenje koje je uspostavljeno između korisnika i *web* stranice. CSRF lažira identitet pošiljatelja zahtjeva tako što navodi njegov *web* preglednik na slanje HTTP zahtjeva u ime korisnika. Za učinkoviti napad koriste se GET i POST HTTP zahtjevi koji izazivaju određene promjene podataka i sadržaja [9].

Prema [9], za uspješno izvršavanje CSRF napada moraju biti ispunjeni sljedeći preduvjeti:

- stranica koja se napada ne provjerava izvor poruke, odnosno HTTP *Referer* zaglavlje
- *web* preglednik korisnika omogućuje lažiranje HTTP *Referer* zaglavlja
- koristi se takav HTTP zahtjev koji izvodi promjene na napadnutoj stranici ili korisničkom računu
- napadač ima pristup autentifikacijskim kolačićima i sigurnosnim značkama preko kojih pristupa ranjivoj stranici
- napadač je u mogućnosti navesti žrtvu na otvaranje zloćudne *web* poveznice

CSRF napadi mogu se izvoditi na različite načine [10]:

- pomoću HTML objekata
- pomoću skriptnog koda umetnutog u HTML
- zlouporabom automatskog generiranja zahtjeva u *web* pregledniku (XMLHttpRequest)

3.3.1. Napadi HTML objektima

CSRF napadi mogu se izvesti na više načina pomoću HTTP GET metode koja služi za dohvat podataka, a jedan od načina je pomoću HTML objekata. Primjer HTML objekta je objekt IMG koji se koristi za oblikovanje slikovnog *web* sadržaja:

**

U atributu "*src*" nalazi se izvor s kojega će se preuzeti slika. Prilikom učitavanja stranice automatski se pristupa tom izvoru kako bi se preuzela slika. Podmetanjem zloćudnog koda u taj atribut moguće je poslati zahtjev drugom *web* odredištu [10]:

**

Učitavanjem *web* stranice na kojoj se nalazi prethodni IMG objekt automatski se pokreće izvršavanje zloćudnog koda na poslužitelju.

Za izvršavanje napada mogu se iskoristiti i HTML objekti IFRAME i SCRIPT koji također sadrže atribut "*src*". IFRAME objekti omogućuju učitavanje drugih *web* stranica u poseban okvir unutar trenutne stranice. SCRIPT objekt sadrži kod ili se kodu udaljeno pristupa preko atributa "*src*". U oba slučaja kao izvor udaljenog koda može se podmetnuti zloćudan zahtjev prema ranjivoj stranici [10]:

<iframe src="http://posluzitelj/zlocudni_kod" />

<script src="http://posluzitelj/zlocudni_kod" />

3.3.2. Napadi skriptnim kodom

Skriptni kod ugniježđen u HTML također može biti sredstvo napada. Ukoliko je omogućeno izvođenje skriptnog koda u *web* pregledniku on se pokreće automatski prilikom učitavanja stranice [10].

3.3.3. Napadi XMLHttpRequest objektom

XMLHttpRequest objekti središnji su objekti AJAX (eng. *Asynchronous JavaScript And XML*) programa koji omogućuje slanje i obradu HTML zahtjeva u pozadinskom radu programa tako da korisnik ne mora čekati ponovno učitavanje stranice. U pozadinskom radu, HTTP zahtjevi se generiraju automatski u *web* pregledniku bez interakcije s korisnikom, a u tu svrhu koristi se XMLHttpRequest objekt [10].

3.3.4. Primjer CSRF napada

Od 2007. zabilježeno je nekoliko slučajeva u kojima je postojala opasnost od CSRF napada [11]:

- *Web* aplikacija za internet bankarstvo banke ING Directa bila je podložna CSRF napadu koji je omogućio neovlaštene prijenose novčanih sredstava

- Poznati *web* servis za gledanje filmova i serija Netflix imao je nekoliko propusta koji su omogućili napadačima izvršavanje radnji kao što su iznajmljivanje DVD-ova u korisnikovo ime, mijenjanje adrese za dostavu pošiljki ili promjenu korisničkih podataka za prijavu
- Popularna *web* stranica YouTube 2008. također je bila podložna CSRF napadu pomoću kojega su napadači mogli izvoditi gotovo sve radnje nekog legitimnog korisnika.

3.4. Kompromitirana autentifikacija i kontrola sjednice

Autentifikacija korisnika koristi se kako bi se utvrdilo da je korisnik uistinu onaj za kojeg se predstavlja da je, a provodi se pomoću korisničkog imena i zaporke. Složenije metode autentifikacije mogu od korisnika zahtijevati neke dodatne informacije poput kriptografskih žetona, ali takvi su mehanizmi skupi za većinu *web* aplikacija [12].

Kada korisnik pristupi *web* stranici uspostavlja se sjednica i svakom se korisniku dodjeljuje jedinstveni identifikator sjednice (eng. *Session ID*). Identifikator sjednice je jedinstveni znakovni niz pomoću kojeg *web* aplikacija razlikuje pojedinog korisnika i prati njegove zahtjeve. Identifikatore sjednice kreira *web* aplikacija te se oni čuvaju i na *web* poslužitelju i na korisnikovom računalu u obliku kolačića (eng. *Cookie*) [12].

3.4.1. Napad na autentifikaciju

Cilj napada na autentifikaciju je otkrivanje podataka o nekom korisniku poput korisničkog imena i zaporke. Jedan od načina otkrivanja podataka o nekom korisniku je korištenje napada grubom silom (eng. *Brute-Force attack*).

Napadač izvodi napad pogađanjem nepoznatih vrijednosti korištenjem svih mogućih kombinacija i analizom dobivenih odgovora. Najveći nedostatak ovakvog napada je vrijeme koje je potrebno da bi se pogodila zaporka korisnika budući modernije *web* aplikacije koriste razne mehanizme za kodiranje zaporki koje mogu biti različitih duljina i kompleksnosti.

3.4.2. Napad na upravljanje sjednicom

Cilj napada na upravljanje sjednicom je preuzimanje identiteta legitimnog korisnika ili otkrivanje i manipulacija zaštićenim podacima vezanim uz sjednicu.

3.4.2.1. Trovanje kolačića

Trovanje kolačića (eng. *Cookie poisoning*) je tehnika manipuliranja ili krivotvorenja kolačića pohranjenih na *web* pregledniku legitimnog korisnika u cilju zaobilaženja sigurnosnih mjera ili slanja lažne informacije poslužitelju. Trovanjem kolačića napadač može dobiti neovlašteni pristup računu legitimnog korisnika na određenoj *web* adresi za koju je kolačić stvoren [13].

3.4.2.2. Krađa identifikatora sjednice

Krađa identifikatora sjednice (eng. *Stealing session ID*) najraširenija je vrsta napada na upravljanje sjednicom. Može se izvršiti korištenjem XSS napada ili prisluškivanjem prometa na mreži u kojoj prijenos podataka između klijenta i poslužitelja nije zaštićen [14].

3.4.2.3. Fiksacija sjednice

Fiksacija sjednice (eng. *Session fixation*) omogućuje napadaču otimanje važeće sjednice legitimnog korisnika. Napad iskorištava ranjivost *web* aplikacije u pogledu upravljanja identifikatorom sjednice. Prilikom autentifikacije korisnika ne dodjeljuje se novi identifikator sjednice, što omogućuje upotrebu postojećeg identifikatora. Napadač mora legitimnom korisniku podmetnuti važeći identifikator i navesti njegov *web* preglednik da ga upotrijebi. Identifikator sjednice napadač može podmetnuti putem URL parametra ili putem skrivene HTML forme [15].

4. ZAŠTITA OD NAPADA

4.1. Zaštita od napada umetanjem SQL koda

Web stranicu moguće je obraniti od većine napada uvođenjem jednostavnih provjera unesenih podataka, korištenjem neuobičajenih imena tablica i atributa u bazi podataka ili tako da se mogućnost izmjene podataka omogući samo korisnicima s administratorskim ovlastima [16].

4.1.1. Polja za unos podataka

Prilikom razvijanja *web* stranice razumno je pretpostaviti da korisničko ime neće biti duže od npr. 10 znakova. Ograničavanjem broja znakova u polju za unos korisničkog imena sprječava se mogućnost unošenja SQL naredbi jer one u pravilu sadrže veliki broj znakova. Također je preporučljivo, gdje je moguće, definirati tip podataka koji se očekuje prilikom unosa. Time se onemogućuje umetanje SQL naredbi u polja koja su namijenjena za unos brojeva i sl.

4.1.2. Baza podataka

Za uspješno izvođenje napada umetanjem SQL koda potrebno je poznavanje određenih parametara o bazi podataka koju se napada poput imena baze, tablica i njenih atributa. Kako bi se napadaču otežalo izvršavanje napada preporuča se korištenje neuobičajenih imena tablica i atributa u bazi podataka. Na primjer, ime tablice koja sadrži popis svih korisnika, kao što je uobičajena praksa, može biti *Users* ili *Korisnici* što napadač može vrlo lako pogoditi. Korištenjem drugačijeg imena tablice, poput *Users_List*, *Lista_Korisnika* i sl., napadaču se otežava napad. Isto vrijedi i za imena atributa u tablicama: umjesto *ime* može se koristiti *korisnicko_ime* ili nešto drugo kako bi se napadaču otežalo pogađanje imena atributa [17].

Također, uvijek se preporuča ograničiti pristup bazi. Obični korisnici trebali bi biti u mogućnosti izvoditi samo SELECT naredbu, a naredbe koje mijenjaju podatke u bazi podataka, poput INSERT, DELETE ili DROP TABLE, trebaju biti omogućene samo korisnicima s administratorskim ovlastima [17].

4.1.3. Provjera podataka

Svi podaci koje korisnik unosi moraju se provjeriti prije prosljeđivanja bazi podataka. Prosljeđivanje podataka bazi podataka bez njihove provjere omogućuje napade umetanjem SQL koda.

Savjeti za provjeru podataka prije njihovog prosljeđivanja bazi podataka su [17]:

- Provjera podataka trebala bi se odvijati na strani poslužitelja. Ako se provjera provodi na strani klijenta, napadač može jednostavno dohvatiti *web* stranicu kao .html datoteku i ukloniti dio koda koji se koristi za provjeru podataka. Nakon toga, napadač može pokretati *web* stranicu lokalno i slati upite bez da se oni provjeravaju.
- Prilagoditi provjeru tipu podataka koji se očekuje. Na primjer, provjera korisničkog imena je drugačija od provjere telefonskog broja budući se telefonski broj sastoji samo od brojeva, dok korisničko ime može imati više različitih znakova. Podaci koji ne odgovaraju očekivanom tipu ne bi se trebali proslijediti bazi podataka.
- Provjera broja znakova. Preveliki broj unesenih znakova može ukazati na potencijalni napad. Na primjer, JMBG korisnika mora sadržavati točno određeni broj znakova, korisničko ime ne smije biti duže od određenog broja znakova itd.
- Provjera navodnika. U većini slučajeva korisnici nemaju potrebu unošenja navodnika, dok napadači koriste navodnike u napadima. Ukoliko se prilikom provjere unesenih podataka otkrije postojanje navodnika, dobra je praksa udvostručiti ih što će rezultirati neispravnim SQL upitom i napadač neće biti u mogućnosti izvršiti napad.
- Provjera znaka točka-zarez (;). Postojanje znaka točka-zarez u unesenim podacima snažno ukazuje na prisutnost programskog koda. Zbog toga se preporuča sve znakove točke-zareza izbaciti iz primljenog niza znakova prije prosljeđivanja upita bazi podataka. Izbacivanjem znaka točka-zarez napadačev SQL kod neće raditi i doći će do greške prilikom izvođenja upita u bazi podataka.
- Provjera dvostrukih crtica (--). U SQL upitima, dvostruke crtice označavaju početak komentara i napadači ih često koriste kako bi izbjegli izvođenje regularnog dijela SQL upita. Preporuča se sve znakove dvostrukih crtica izbaciti iz primljenog niza znakova prije prosljeđivanja upita bazi podataka. Izbacivanje dvostrukih crtica napadačev SQL kod neće raditi i doći će do greške prilikom izvođenja upita.
- Provjera postojanja SQL ključnih riječi. Ključne riječi poput SELECT, DELETE, INSERT ili DROP TABLE mogu ukazati na potencijalni napad. Napadač može maskirati

svoj SQL kod tako da ne koristi SQL ključne riječi. Na primjer, ključna riječ SHUTDOWN koja gasi bazu podataka može se zapisati na sljedeći način: exec(char(0x73687574646f776e)). Niz brojeva koji funkcija *char()* prima kao argument zapravo je heksadecimalni ASCII zapis riječi SHUTDOWN, a funkcija *char()* primljeni heksadecimalni zapis pretvara u niz znakova (riječ SHUTDOWN). Zbog ovakvih slučajeva, provjera postojanja SQL ključnih riječi ne može sama osigurati od napada umetanjem SQL koda.

4.1.4. Poslužitelj

Programu koji se nalazi na poslužitelju treba dodijeliti minimum ovlasti, onoliko koliko je dovoljno da obavlja svoj rad. Kako bi se napadaču otežalo prikupljanje informacija o bazi podataka, potrebno je ograničiti informacije dostupne u porukama greške. Preporuča se preusmjeravanje korisnika na stranicu koja navodi da je došlo do greške, ali ne pruža detalje zbog čega je do greške došlo [17].

Poslužitelj je stalno potrebno nadgledati, provjeravati i nadograđivati najnovijim programskim nadogradnjama. Također, poželjno je raditi sigurnosne kopije podataka (eng. *backup*) [17].

4.1.5. Pripremljene naredbe

Pripremljene naredbe (eng. *Prepared Statements*) omogućuju da se unaprijed definiraju SQL upiti i izvršavaju s parametrima koji se dinamički dodaju.

Koraci u korištenju pripremljenih naredbi su:

- kreiranje objekta za spajanje na bazu:
  - *$mysqli = new mysqli("localhost", "root", "zaporka123", "DBime");*
- kreiranje SQL upita bez konkretnih parametara umjesto kojih se koristi znak upita (?):
  - *$query = "SELECT telefon FROM users WHERE ime = ? AND zaporka = ?;";*
- dohvaćanje unesenih podataka:
  - *$name = $_POST['ime'];*
  - *$password = $_POST['zaporka'];*
- kreiranje instance naredbe i priprema upita:

- *$statement = $mysqli->stmt_init();*
- *$statement->prepare($query);*

- željeni parametri nadodaju se na pripremljeni upit, prilikom čega se definira i tip podatka koji se prosljeđuje (npr. i – cijeli broj, s – znakovni niz):
  - *$statement->bind_param('s', $name);*
  - *$statement->bind_param('s', $password);*
- Na kraju je potrebno izvršiti upit nad bazom podataka, dohvatiti rezultate te zatvoriti konekciju na bazu:
  - *$statement->execute();*
  - *$statement->bind_result($tel);*
  - *$statement->close();*
  - *$mysqli->close();*

4.2. Zaštita od napada umetanjem skriptnog koda

Razvojni programeri mogu zaštiti *web* stranicu od neželjenih napada tako da osiguraju da dinamički generirana stranica ne sadrži neželjene HTML oznake (eng. *tags*) što se postiže filtriranjem i kodiranjem *web* sadržaja. Korisnik se može zaštiti onemogućavanjem izvođenja skriptnih jezika u svom *web* pregledniku što pruža najbolju zaštitu, ali za posljedicu ima smanjenje funkcionalnosti [18].

Svi ulazni podaci predstavljaju potencijalnu prijetnju. Stoga je potrebno izvršiti filtriranje posebnih HTML znakova te ih kodirati kako bi se izbjeglo prezentiranje HTML-a u korisnikovom *web* pregledniku. U tablici 4.1. prikazani su posebni HTML znakovi i njihovi odgovarajući kodovi.

Tab. 4.1. *Posebni HTML znakovi i njihovi kodovi*

HTML znak	Odgovarajući kod
&	&amp
„	&quot
'	&#x27
<	&lt
>	&gt
/	&#x2F

Osim posebnih HTML znakova potrebno je provjeriti postoje li oznake skriptnog jezika (eng. *script tags*) u ulaznim poljima i parametrima poveznica.

Filtriranje ulaznih podataka nije dovoljno učinkovito budući se dinamički sadržaj može spremiti u bazu podataka *web* stranice koristeći razne metode, a ne samo koristeći HTTP. U ovom slučaju poslužitelj neće vidjeti podatke kao dio procesa ulaza podataka te će oni ostati zaraženi. Preporučuje se filtriranje sadržaja uključiti u izlazni proces tj. filtrirati podatke prije njihovog uključivanja u dinamičku *web* stranicu [18].

Zaštita od napada umetanjem skriptnog jezika može se postići korištenjem seta funkcija koje nudi PHP (eng. *Hypertext Preprocessor*) programski jezik, namijenjen razvoju dinamičkih *web* stranica.

Jedna od funkcija je *htmlspecialchars()* koja kao parametar prima izvorni ulaz od korisnika u obliku znakovnog niza i pretvara posebne HTML znakove u njihove odgovarajuće kodove:

```
<?php

$ulaz = "<script>alert('XSS napad!')</script> ";   #ulaz od strane napadača

$izlaz = htmlspecialchars($ulaz);     #obrada ulaza funkcijom htmlspecialchars()

echo $izlaz;     #izlaz: &lt;script&gt;alert('XSS napad!')&lt;/script&gt;

?>
```

Druga funkcija koja se također može koristiti u obrani od napada je funkcija *strip_tags()*. Funkcija kao parametar prima ulaz od korisnika u obliku znakovnog niza, a kao rezultat vraća znakovni niz bez HTML oznaka:

```
<?php

$ulaz = "<script>alert('XSS napad!')</script> ";   #ulaz od strane napadača

$izlaz = strip_tags($ulaz);     #obrada ulaza funkcijom strip_tags()

echo $izlaz;     #izlaz: alert('XSS napad!')

?>
```

4.3. Zaštita od CSRF napada

Zaštita od CSRF napada zasniva se na dvije provjere [19]:

- Provjera standardnih HTTP zaglavlja
- Provjera CSRF žetona

4.3.1. Provjera standardnih HTTP zaglavlja

HTTP zaglavlja (eng. *HTTP Headers*) su komponente poruka zahtjeva (eng. *request*) i odgovora (eng. *response*) koje definiraju parametre jedne HTTP transakcije.

Provjera standardnih HTTP zaglavlja sastoji se od dva koraka:

1. Utvrđivanje lokacije s koje je poslan zahtjev (eng. *source origin*)
2. Utvrđivanje lokacije na koju se šalje zahtjev (eng. *target origin*)

Iako je vrlo lako manipulirati vrijednostima HTTP zaglavlja korištenjem JavaScript programskog koda, neke je nemoguće mijenjati u žrtvinom *web* pregledniku tijekom CSRF napada. Vrijednost takvih HTTP zaglavlja može mijenjati samo *web* preglednik što ih čini vrlo pouzdanima budući se njihova vrijednost ne može mijenjati niti XSS napadom [19].

4.3.1.1. Utvrđivanje lokacije s koje je poslan zahtjev

Za utvrđivanje lokacije s koje je poslan zahtjev preporuča se provjera jednog od sljedećih HTTP zaglavlja koja su uključena u gotovo svaki zahtjev:

- *Origin* zaglavlje
- *Referer* zaglavlje

Ako u zahtjevu postoji *Origin* zaglavlje, njegova vrijednost mora odgovarati vrijednosti lokacije na koju se šalje zahtjev. Za razliku od *Referer* zaglavlja, *Origin* zaglavlje će biti uključeno u HTTP zahtjeve koji potječu s HTTPS (eng. *HTTP Secured*) URL-a [19].

Ako u zahtjevu ne postoji *Origin* zaglavlje provjerava se *Referer* zaglavlje. Provjera *Referer* zaglavlja uobičajena je metoda za obranu od CSRF napada u ugrađenim mrežnim uređajima gdje

ne postoji potreba za postojanje korisničkog stanja. Ova metoda obrane od CSRF napada također se upotrebljava u radu s neautoriziranim zahtjevima kao što su zahtjevi poslani prije uspostavljanja sjednice [19].

Rijetko se može dogoditi da u zahtjevu ne postoje niti *Origin* niti *Referer* zaglavlje. U ovom slučaju zahtjev se može prihvatiti ili odbaciti. Ukoliko se kao dodatni korak zaštite ne radi provjera CSRF žetona, preporučljivo je odbaciti zahtjev.

4.3.1.2. Utvrđivanje lokacije na koju se šalje zahtjev

Utvrđivanje lokacije na koju se šalje zahtjev nije jednostavan zadatak budući se glavni poslužitelj često nalazi iza jednog ili više proxy poslužitelja[1], te se izvorni URL razlikuje od URL-a koji glavni poslužitelj primi.

Za utvrđivanje lokacije na koju se šalje zahtjev može se koristiti *Host* zaglavlje čija je vrijednost lokacija na koju se šalje zahtjev. Ali, ako se koriste proxy poslužitelji, tada će svaki proxy poslužitelj promijeniti lokaciju u *Host* zaglavlju na URL adresu sljedećeg poslužitelja. Vrijednost *Host* zaglavlja razlikovat će se od vrijednosti u *Origin* i *Referer* zaglavlju.

Kako bi se sačuvala izvorna vrijednost *Host* zaglavlja, koristi se zaglavlje *X-Forwarded-Host*. Proxy poslužitelji prosljeđuju zaglavlje bez mijenjanja njegove vrijednosti te se ono može iskoristiti za utvrđivanje lokacije [19].

4.3.2. Provjera CSRF žetona

Nakon provjere standardnih HTTP zaglavlja, kao dodatna mjera sigurnosti, provjerava se vrijednost CSRF žetona (eng. *CSRF token*) koji *web* stranica ugrađuje u formu ili URL. Žeton je jednokratna vrijednost koja se stvara kada korisnik posjeti stranicu, a koristi se za provjeru je li korisnik uistinu onaj koji stvara zahtjev.

[1] Proxy poslužitelj – računalo koje stoji između klijenta i glavnog poslužitelja kao posrednik

CSRF žeton se u formu ugrađuje u obliku sakrivenog polja:

```
<form method="POST" action="/spremi-podatke">
    <!-- Sakriveno polje u kojem se nalazi vrijednost CSRF žetona -->
    <input type="hidden" name="_token" value="QNZ6nuWpKtlYWEwYzU1YWQwMT">
    ...
</form>
```

CSRF žeton se kreira samo jednom za trenutnu sjednicu. Nakon što se generira, žeton se sprema u sjednicu i koristi se za svaki HTTP zahtjev dok sjednica ne istekne. Kada korisnik pošalje zahtjev poslužitelj mora provjeriti postojanje i vrijednost žetona. Ukoliko žeton nije pronađen ili se njegova vrijednost razlikuje od vrijednosti spremljene u sjednici, zahtjev se odbacuje. Žeton je potrebno ponovno generirati i zapisati zahtjev kao potencijalni CSRF napad [19].

4.4. Zaštita od napada na autentifikaciju i upravljanje sjednicom

Pažljiva i pravilna upotreba mehanizama za autentifikaciju i upravljanje sjednicom trebala bi značajno smanjiti vjerojatnost napada. Definiranje i dokumentiranje pravila *web* lokacije za sigurno upravljanje korisnikovim podacima dobar je korak ka zaštiti od napada.

Savjeti za zaštitu [20]:

- **Jakost zaporke**
    - Zaporke trebaju zahtijevati minimalnu duljinu i biti dovoljno kompleksne. Kompleksnost zahtjeva minimalnu kombinaciju abecednih, numeričkih i/ili ne-alfanumeričkih znakova u korisničkoj zaporci. Zaporke je potrebno periodično mijenjati i onemogućiti ponovnu upotrebu prethodnih zaporki.
- **Upotreba zaporke**
    - Korisnicima mora biti ograničen broj pokušaja prijave u sustav u jedinici vremena, a neuspjeli pokušaji prijave moraju se zapisati u sustav. Sustav ne bi trebao naznačiti je li pogrešno korisničko ime ili zaporka u slučaju neuspjelog pokušaja

prijave. Korisnici bi trebali biti obavješteni o datumu i vremenu njihove posljednje uspješne prijave u sustav i o broju neuspjelih pokušaja od tog vremena.

- **Kontrole za promjenu zaporke**
    - Mehanizam za promjenu zaporke treba upotrijebiti gdje god je korisnicima dopušteno mijenjanje zaporke. Prilikom promjene zaporke, od korisnika se mora zahtijevati i stara i nova zaporka. Ako se zaboravljene zaporke šalju korisnicima e-poštom, sustav od korisnika mora zahtijevati ponovnu autentifikaciju pri svakoj promjeni adrese e-pošte. U suprotnom napadač, koji privremeno ima pristup sjednici legitimnog korisnika, može promijeniti adresu e-pošte i zatražiti da se zaboravljena zaporka pošalje njemu.
- **Pohrana zaporke**
    - Sve zaporke moraju se spremati u šifriranom obliku. Ključ za dešifriranje zaporke mora se dobro zaštititi kako bi se spriječila njihova krađa. Ukoliko napadač dođe u posjed ključa za dešifriranje, može vrlo lako otkriti sve zaporke spremljene u sustavu.
- **Zaštita podataka u prijenosu mrežom**
    - Jedina učinkovita tehnika za zaštitu podataka u prijenosu je šifriranje cijele transakcije prijave pomoću SSL-a[2] (eng. *Secure Socket Layer*).
- **Zaštita identifikatora sjednice**
    - U idealnom slučaju, sjednica će bit zaštićena SSL-om, a identifikator sjednice neće se moći dohvatiti s mreže. Ukoliko SSL nije dostupan, identifikator sjednice je potrebno zaštititi na druge načine. Identifikatori sjednice ne bi trebali biti uključeni u URL. Trebaju biti dugi, složeni slučajni brojevi koji se ne mogu jednostavno pogoditi. Identifikatori koje je odabrao korisnik nikada se ne smiju prihvatiti kao važeći.
- **Popis korisničkih računa**
    - Sustavi trebaju biti osmišljeni tako da se korisnicima onemogući pristup popisu naziva korisničkih računa na *web* aplikaciji. Ukoliko popis naziva mora biti dostupan, preporuča se umjesto pravoga naziva koristiti pseudonim koji je poveznica na stvarni račun. Na taj se način pseudonim ne može koristiti pri pokušaju prijave u sustav.

[2] SSL – protokol transportnog sloja koji omogućava sigurnu komunikaciju preko interneta

5. ALATI ZA TESTIRANJE SIGURNOSTI WEB APLIKACIJA

Alati za testiranje sigurnosti *web* aplikacija mogu raditi na dva principa. Prvi jest usporedba ponašanja stranice (zahtjeva i odgovora) sa poznatim obrascima ranjivosti, a drugi je nasumično ispitivanje tj. generiranje nasumičnih ulaznih parametara za aplikaciju uz praćenje ponašanja aplikacije uz dane parametre. Prvi način često se naziva i *glass-box* ispitivanje, jer su korisniku vidljivi svi procesi i obrasci koji se testiraju, dok se drugi način naziva *black-box* ispitivanje u kojemu su ulazi i izlazi nepoznati korisniku [21].

5.1. Nikto alat

Nikto alat pripada prvoj skupini alata, tj. skupini koja uspoređuje ponašanje *web* aplikacije sa obrascima poznatih ranjivosti. Temelji se na Perl programskom jeziku, a može se pokrenuti na većini operacijskih sustava s instaliranim Perl prevoditeljem (eng. *interpreter*) i bilo kojim *web* poslužiteljem (*Apache, Nginx, IHS, OHS, Litespeed,* itd.) [22].

Nikto se pokreće iz komandne linije i ima mogućnost pronalaska:

- Uobičajenih loših postavki izvršnog okruženja,
- Nesigurnih datoteka te
- Neažuriranih inačica poslužitelja i biblioteka

Naredba za pokretanje izvršavanja ispitivanja ranjivosti jest "*perl nikto.pl –h hostname*" gdje je *hostname* IP ili URL adresa *web* aplikacije koja se testira:

perl nikto.pl –h 192.168.0.1

Ova naredba će pokrenuti izvršavanje ispitivanja ranjivosti na IP adresi 192.168.0.1. Ukoliko se ne postavi drugačije, ispitivanje ranjivosti će se izvršavati na portu 80. Za ispitivanje ranjivosti na drugom portu, koristi se opcija "*–p port*":

perl nikto.pl –h 192.168.0.1 –p 443

Poslužitelj, port i protokol mogu se definirati korištenjem kompletnog URL-a:

perl nikto.pl –h https://192.168.0.1:443/

Ako se želi izvršiti ispitivanje ranjivosti na više portova, npr. 443, 80 i 88, koristi se sljedeća naredba:

perl nikto.pl –h 192.168.0.1 –p 443,80,88

Nikto alat nakon pokretanja započinje ispitivanje te u stvarnom vremenu prijavljuje pronađene nedostatke i ranjivosti aplikacije.

5.2. Wapiti alat

Wapiti alat pripada drugoj skupini, tj. skupini alata koji nasumično generiraju upite, šalju ih *web* aplikaciji te analiziraju odgovor. Wapiti alat teži k prepoznavanju ranjivosti koje nisu dokumentirane i definirane.

Alat je u mogućnosti otkriti većinu ranjivosti umetanjem parametara u tijelo CGI (eng. *Common Gateway Interface*) skripte[3] koja se izvršava. Aplikacija odgovara na svaki oblik korisničkog unosa dok Wapiti promatra odgovore aplikacije. Alat uspoređuje stvarno ponašanje aplikacije s definiranim, te izvještava korisnika o mogućoj ranjivosti [21].

Wapiti se pokreće u komandnoj liniji naredbom "*wapiti.py hostname*" gdje je *hostname* URL adresa *web* aplikacije koja se testira:

wapiti.py http://www.hostname.com/

Alat je u mogućnosti otkriti:

- Greške u rukovanju datotekama
- Napade umetanjem SQL koda
- XSS napade

[3] CGI skripta – programski kod koji se izvršava na *web* poslužitelju

6. ZAKLJUČAK

Zadatak diplomskog rada bio je opisati najčešće napade na *web* aplikacije te navesti postupke zaštite i primjere napada.

Najčešći napadi koji se izvode su: napad umetanjem SQL koda, napad umetanjem skriptnog koda, napad krivotvorenjem zahtjeva, napad na autentifikaciju i napad na sjednicu. Uspješnim izvršavanjem napada napadač može doći u posjed povjerljivih informacija.

Napad umetanjem SQL koda izvodi se pomoću polja za unos podataka ili URL-a. Za smanjenje opasnosti od napada potrebno je uvesti provjeru podataka koje korisnik želi unijeti u bazu podataka. Također je dobra praksa koristiti neuobičajene nazive baze podataka i tablica u bazi, ograničiti prava pristupa korisnika i ne prikazivati korisnicima detaljne poruke greške.

Postoje tri tipa napada umetanjem skriptnog koda: trajni, jednokratni i DOM bazirani. Trajni napad se izvodi trajnim umetanjem skriptnog koda na *web* stranicu, jednokratni se izvodi tako da se žrtvu navede da posjeti stranicu na kojoj je omogućeno izvršavanje koda, a DOM bazirani napad se, za razliku od trajnog i jednokratnog, izvodi na žrtvinom računalu. Za zaštitu od napada potrebno je filtrirati posebne HTML znakove u dinamičkim stranicama koje korisnici posjećuju.

Napad krivotvorenjem zahtjeva iskorištava povjerenje koje *web* stranica ima prema korisniku. Ovim se napadom lažira identitet pošiljatelja zahtjeva, a za izvršavanje se koriste GET i POST HTTP zahtjevi. Provjerom standardnih HTTP zaglavlja i CSRF žetona smanjuje se vjerojatnost napada.

Napad na autentifikaciju i sjednicu izvode se s ciljem krađe identiteta legitimnog korisnika ili otkrivanja i manipulacije zaštićenim podacima vezanim uz sjednicu. Napad na sjednicu izvodi se: trovanjem kolačića, krađom identifikatora sjednice i fiksacijom sjednice. Određenu razinu zaštite od napada moguće je postići pažljivim i pravilnim korištenjem mehanizama za autentifikaciju i upravljanje sjednicom.

LITERATURA

[1] https://msdn.microsoft.com/en-us/library/ff648636.aspx
pristup ostvaren (10.3.2018.)

[2] Laboratorij za sustave i signale, Napadi umetanjem SQL koda, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, Zagreb, 2011.

[3] https://en.wikipedia.org/wiki/SQL_injection#Examples
pristup ostvaren (15.3.2018.)

[4] http://www.aquilonis.hr/acunetix/cross_site_scripting.html
pristup ostvaren (28.5.2017.)

[5] http://www.cert.hr/sites/default/files/CCERT-PUBDOC-2006-05-157.pdf
pristup ostvaren (28.5.2017.)

[6] https://www.incapsula.com/web-application-security/cross-site-scripting-xss-attacks.html
pristup ostvaren (30.5.2017.)

[7] https://www.incapsula.com/web-application-security/reflected-xss-attacks.html
pristup ostvaren (30.5.2017.)

[8] https://en.wikipedia.org/wiki/Samy_(computer_worm)
pristup ostvaren (15.3.2018.)

[9] http://www.cert.hr/sites/default/files/NCERT-PUBDOC-2010-04-297.pdf
pristup ostvaren (01.06.2017.)

[10] https://www.cis.hr/www.edicija/LinkedDocuments/NCERT-PUBDOC-2010-04-297.pdf
pristup ostvaren (02.03.2018.)

[11] https://en.wikipedia.org/wiki/Cross-site_request_forgery#History
pristup ostvaren (15.3.2018.)

[12] https://www.owasp.org/index.php/Broken_Authentication_and_Session_Management#Description
pristup ostvaren (12.3.2018.)

[13] https://security.radware.com/ddos-knowledge-center/ddospedia/cookie-poisoning/
pristup ostvaren (12.3.2018.)

[14] https://en.wikipedia.org/wiki/Session_hijacking#Methods
pristup ostvaren (12.3.2018.)

[15] https://www.owasp.org/index.php/Session_fixation
pristup ostvaren (12.3.2018.)

[16] https://www.cis.hr/dokumenti/napadiumetanjemsqlkoda.html

pristup ostvaren (03.03.2018.)

[17] https://www.cis.hr/files/dokumenti/CIS-DOC-2011-09-025.pdf
pristup ostvaren (03.03.2018.)

[18] https://www.ibm.com/developerworks/tivoli/library/s-csscript/
pristup ostvaren (05.03.2018.)

[19] https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet
pristup ostvaren (05.03.2018.)

[20] https://www.owasp.org/index.php/Broken_Authentication_and_Session_Management#How_to_Protect_Yourself
pristup ostvaren (15.3.2018.)

[21] https://www.cis.hr/sigurosni-alati/ispitivanje-ranjivosti-web-aplikacija.html
pristup ostvaren (05.03.2018.)

[22] https://geekflare.com/nikto-webserver-scanner/
pristup ostvaren (05.03.2018.)

SAŽETAK

Sigurnost je stupanj zaštite imovine od opasnosti, štete, gubitka ili kriminalne aktivnosti. Osnovni elementi sigurnosti su: autentifikacija, autorizacija, provjera, povjerljivost, integritet i dostupnost.

Propusti u razvoju programskih rješenja narušavaju sigurnost *web* aplikacija. Najčešći napadi koji se izvode su: napad umetanjem SQL koda, napad umetanjem skriptnog koda (XSS napad), napad krivotvorenjem zahtjeva (CSRF napad) i napad na autentifikaciju i upravljanje sjednicom.

Za uspješnu obranu od napada potrebno je razviti mehanizme koji će provjeravati, filtrirati i zaštititi podatke koje korisnik i *web* aplikacija međusobno razmjenjuju.

Ključne riječi: sigurnost, napad, zaštita, baza podataka, autentifikacija, HTTP, HTML, JavaScript, SQL, XSS, CSRF

SUMMARY

WEB APPLICATION PROTECTION PROCEDURES

Security is a degree of protection of property from danger, damage, loss or criminal activity. The fundamental elements of security are: authentication, authorization, auditing, confidentiality, integrity and availability.

Flaws in the development of software solutions undermine the security of web application. The most frequent attacks are: SQL injection attatck, cross-site scripting attack (XSS attack), cross-site request forgery attack (CSRF attack), and broken authentication and session management.

For successful protection against attacks, it is necessary to develop mechanisms that will validate, filter, and protect the data that users exchange with web application.

Keywords: security, attack, protection, database, authentication, HTTP, HTML, JavaScript, SQL, XSS, CSRF

VEDRAN GLAVAŠ

Dubrovačka 9

31326 Darda

Mobitel: 091 602 3749

e-mail: vglavas93@gmail.com

Ž I V O T O P I S

Vedran Glavaš rođen je 2. listopada 1993. u Osijeku, Republika Hrvatska, od oca Davora i majke Sande, rođene Vučak.

Prvi razred osnovne škole pohađao je u Osijeku u Osnovnoj školi Sv. Ane, dok je ostale razrede pohađao u Osnovnoj školi Darda u Dardi gdje je 2008. i završio osnovno školovanje s odličnim uspjehom.

Srednju Elektrotehničku i prometnu školu u Osijeku pohađao je u vremenu od 2008. do 2012. te stekao zvanje elektrotehničar.

Sveučilišni preddiplomski studij elektrotehnike, smjer Komunikacije i informatika, na Elektrotehničkom fakultetu Osijek uspješno je završio 2015. s temom završnog rada "Karakterizacija različitih tehnologija baterija".

Trenutno je student 2. godine sveučilišnog diplomskog studija elektrotehnike, smjer Komunikacije i informatika, modul DKB – Mrežne tehnologije, na Sveučilištu Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek.

Posjeduje vozačku dozvolu s položenom B kategorijom. Odlično se služi engleskim jezikom i u govoru i u pismu. Veliki je zaljubljenik u sve vrste tehnike, posebice računala i mobitela. Posjeduje visok stupanj informatičkog znanja i sposobnosti održavanja računala i računalne opreme, te korištenja računalnih aplikacija Microsoft Office programskog paketa. Poznaje sljedeće programske jezike: Java, PHP, JavaScript, SQL, HTML i CSS.

Vedran Glavaš