ZAVOD ZA ELEKTRONIKU, MIKROELEKTRONIKU, RAČUNALNE I INTELIGENTNE SUSTAVE
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
SVEUČILIŠTE U ZAGREBU

DIPLOMSKI RAD br. 1685

Automatizirano određivanje vrste Web aplikacije

Mario Kozina

Zagreb, rujan 2007.

Sažetak

U ovom radu se prvenstveno opisuje postupak automatiziranog otkrivanja vrste Web aplikacije koji predstavlja ključan korak u otkrivanju poznatih ranjivosti u Web aplikacijama. U radu su također opisani problemi sigurnosti i ranjivosti Web aplikacija, te principi ispitivanja sigurnosti Web aplikacija korištenjem različitih alata otvorenog koda. Radi lakšeg razumijevanja, većinu opisa prate pripadajući primjeri. U primjerima se prikazuju različite napadače tehnike, analize napada, korištenje alata za ispitivanje sigurnosti, te rad sustava za ispitivanje poznatih sigurnosti rupa u Web aplikacijama. U praktičnom dijelu ostvaren je programski okvir za sustav za ispitivanje poznatih ranjivosti u Web aplikacijama, a posebno je razrađen i ispitan sustav za otkrivanje vrste Web aplikacije.

Abstract

This diploma thesis describes technique of automatic detection of Web application type, which is an important step in detection of known vulnerabilites in Web applications. Diploma thesis also describes security problems, vulnerabilities of Web applications and principles of testing Web applications by using various open-source tools. For easier understanding, theoretical descriptions are followed by examples. Examples show various attack techniques, attack analysis, usage of tools for testing Web application security and funcionality of the system which detects known security holes in Web applications. Pratical part includes framework which is used to detect know vulnerabilites in Web applications, in which is specially elaborated part which detects Web application type.

Sadržaj

1. Uvod

Pojava i razvoj Interneta krajem osamdesetih i početkom devedesetih godina prošlog stoljeća omogućili su korisnicima globalni pristup različitim tipovima informacija i nevjerojatne mogućnosti pri njihovoj razmjeni. *Word Wide Web* ili kraće *Web,* je najrašireniji sustav koji koristi svjetsku mrežu Internet za pristup informacijama. Sustav Web-a se temelji na međusobno povezanim dokumentima kojima se može pristupiti kroz komunikacijsku infrastrukturu Interneta. Web dokumenti poznati kao Web stranice sadrže tekstualni sadržaj, slike i drugi multimedijalni sadržaj. Web stranice također sadrže poveznice koje omogućavaju korisniku navigaciju na druge Web stranice. Web se temelji na klijentsko - poslužiteljskoj arhitekturi. Korisnik putem URL-a zahtijeva određenu Web stranicu, Web preglednik stvara HTTP zahtjev prema Web poslužitelju, te poslužitelj odgovara na zahtjev sadržajem opisanim HTML jezikom koji Web preglednik prilagođava i prikazuje korisniku u obliku Web stranice.

U samim počecima Web-a, govori se o statičnim Web stranicama. Statične Web stranice uvijek sadrže iste informacije, neovisno o zahtjevu korisnika. Drugim riječima, kada Web poslužitelj primi zahtjev s URL-om statične Web stranice, on će uvijek vratiti isti sadržaj u HTML obliku, neovisno o kontekstu i korisniku. Problem statičnih Web stranica je nemogućnost interakcije s korisnikom. Naime, statična Web stranica ne može primiti korisničke podatke te provesti radnje prilagođene samom korisniku. Prema tome, nije moguća kontrola konteksta, provedba autentifikacijskih i autorizacijskih procesa ni izrada sadržaja prilagođena određenom korisniku. Sredinom devedesetih godina dolazi do pojave prvih dinamičkih Web stranica koje su pružile korisnicima interaktivno iskustvo. Sadržaj dinamičke Web stranice se mijenjao i prilagođavao u ovisnosti o kontekstu i o uvjetima definiranim od strane korisnika. Interaktivnost se postigla korištenjem skripti koje se nalaze na korisničkoj i poslužiteljskoj strani. Skripte na korisničkoj strani su poboljšale mogućnost prezentacije Web stranica, dok su skripte na poslužiteljskoj strani omogućile procesiranje korisničkih podataka te provođenje složenijih procesa poput autentifikacije i pristupa bazi podataka. Uvođenjem dinamike u Web, više se ne govori o Web stranicama, već o Web aplikacijama.

U današnje vrijeme Web aplikacije su dominantna i najsofisticiranija tehnologija korištena u različitim poslovima na Internetu. Njihovo ubrzano širenje je polučeno uslugama koje omogućuju korisnicima korištenje, razmjenu i promjenu informacija neovisno o platformi kroz infrastrukturu Interneta. Primjeri Web aplikacija su pretraživači, Web-mail aplikacije, aplikacije za kupnju i različiti portali. Upravo zbog njihove raširenosti, dostupnosti i načina na koji rukuju informacijama, Web aplikacije su postale kritične sigurnosne točke modernog Interneta.

Većina razvojnih okruženja Web aplikacija daje na korištenje podatkovne elemente programeru, pri čemu izostavljaju pojedinosti o načinu na koji se ti podatkovni elementi interpretiraju, odnosno ne provjerava se smisao podataka unesenih od strane korisnika (engl. *validation and sanity checking*). Na primjer, često se ne provjeravaju korisnički podaci unešeni korištenjem polja-elementa za unos teksta (engl. *textbox*). Kao rezultat takvog pristupa i nesmotrenosti Web programera, dolazi do pojave različitih vrsta ranjivosti unutar Web aplikacija. Identificiranje ranjivosti omogućava potencijalnom napadaču korištenje različitih napadačkih tehnika. Korištenjem napadačkih tehnika može se ugroziti funkcionalnost Web aplikacije, narušiti procese autentifikacije i autorizacije, onemogućiti dostupnost usluga Web aplikacije ili čak izbrisati interne podatke Web aplikacije.

Pojam sigurnosti Web aplikacija je u uskoj vezi s postupkom otkrivanja ranjivosti unutar Web aplikacija. Danas postoji mnoštvo komercijalnih alata i alata otvorenog koda koji se koriste u ove svrhe. Cilj alata koji se koriste za ispitivanje sigurnosti je ručno ili automatizirano otkrivanje poznatih ili nepoznatih ranjivosti unutar određene Web aplikacije. Iako su komercijalni alati moćniji i imaju veći bazu ranjivosti, zbog svoje cijene, većini korisnika su neprihvatljivi. Stoga, posebna pozornost u ovom radu će se posvetiti alatima otvorenog koda.

U okviru ovog rada promatrati će se sustav specijaliziran za automatizirano traženje poznatih ranjivosti u Web aplikacijama. Kako je automatizirano traženje ranjivosti složeni proces koji se odvija u više faza, sustav će se modularno ostvariti tako da funkcionalnost pojedinog podsustava (modula) odgovara pojedinoj fazi procesa traženja ranjivosti. Tri su glavne faze u tom procesu:

- prikupljanje korisnih informacija,
- određivanje vrste Web aplikacije i
- korištenje napadačke tehnike i analiza uspješnosti.

Poseban naglasak ovog rada je na podsustavu za određivanje vrste Web aplikacije. Budući da današnji alati uglavnom otkrivaju vrstu Web poslužitelja, te vrstu i verziju operacijskog sustava na kojem je pokrenuta Web aplikacija, pojavila se ideja za programskim ostvarenjem alata koji će otkriti vrstu Web aplikacije u svrhu pojednostavljenja procesa o ranjivosti.

Struktura diplomskog rada je sljedeća. U drugom poglavlju se razmatraju statistički pokazatelji ranjivosti Web aplikacija, opisuju se ranjivosti i napadačke tehnike vezane uz Web aplikacije, te je dan kratak pregled mogućih zaštita i mogućnosti prevencije napada. U trećem poglavlju se opisuju tehnike ispitivanje sigurnosti Web aplikacija korištenjem alata, te se opisuje par alata otvorenog koda koji se koriste u te svrhe. Četvrto poglavlje sadrži detaljan opis sustava za automatizirano traženje poznatih ranjivosti, s naglaskom na podsustav za otkrivanje vrste Web aplikacije. U petom poglavlju se opisuje ispitivanje sustava za otkrivanje vrste Web aplikacije te se razmatraju i komentiraju rezultati.

2. Sigurnost Web aplikacija i podjela ranjivosti

Sigurnost Web aplikacija je postala goruć problem unutar cjelokupne sigurnosti Interneta. U zadnje dvije godine čak više od 50% svih ranjivosti u CVE (*Common Vulnerabilities an Exposures*) bazi su ranjivosti Web aplikacija [3.]. Tablica 2.1 prikazuje i klasificira razinu sigurnosti arhitekture određenih kategorija Web aplikacija i programske podrške potrebne za njihov rad.

Tablica 2.1. Razina sigurnosti kategorija Web aplikacija i programske podrške

1. Korisničke Web aplikacije	
2. Komercijalne Web aplikacije i Web aplikacije otvorenog koda	
3. Web poslužitelj (*Apache* ili *Microsoft IIS*)	
4. Baza podataka (*Oracle, MySQL, DB2)*	
5. Komercijalne aplikacije i aplikacije otvorenog koda	
6. Operacijski sustav (*Windows, Linux, OS X*)	
7. Mreža (vatrozid, preusmjerivač)	

Vidljivo je da su najugroženije i najnesigurnije Web aplikacije koje su posebno izrađene za određenog korisnika po njegovoj specifikaciji. Razlog tomu je zanemarivanje sigurnosnih aspekata od strane Web programera i dizajnera. Većina programera se orijentira na ostvarenje funkcionalnosti Web aplikacije i poštivanje rokova izrade kako bi ispunili svoje obveze prema korisniku. Također, većina programera nije dovoljno educirana i nije svjesna svojih propusta u programskom kodu koji mogu uzrokovati određenu vrstu ranjivosti u Web aplikaciji.

Manje ugrožene su Web aplikacije otvorenog koda i komercijalne Web aplikacije. Razlozi ranjivosti su slični onima kao kod posebno izrađenih Web aplikacija. Međutim, ove vrste Web aplikacija su široko raširene, pa educiraniji korisnici mogu aktivno sudjelovati u podizanju razine sigurnosti. Primjerice, korisnici mogu otkriti određenu ranjivost i izvijestiti programera Web aplikacija. Programer će intervenirati izmjenom u kodu, te će izdati sigurnosnu zakrpu kako bi eliminirao prijavljene ranjivosti.

Web poslužitelji su u uskoj vezi s radom Web aplikacija. Najdostupniji su, pa su i najugroženiji unutar programske podrške potrebne za rad Web aplikacija. Stoga, potencijalni napadač često posvećuje pažnju identifikaciji i ugrožavanju Web poslužitelja na kojem je upogonjena određena Web aplikacija. Napadima se želi prouzrokovati nedostupnost Web poslužitelja, odnosno Web aplikacija koje su na njemu upogonjene. Baze, lokalne aplikacije i operacijski sustavi su manje ugroženi od Web poslužitelja, ali i dalje postoji opasnost njihova ugrožavanja korištenjem posebnih napada poput ubacivanja SQL koda i OS naredbi. Mrežni elementi su najsigurniji iz razloga što su mrežni napadi znatno kompleksniji od aplikacijskih.

Sve više organizacija se bavi problemom sigurnosti Web aplikacija. Dvije vodeće međunarodne organizacije koje su posvetile svoj rad upravo ovom području sigurnosti su WASC – Web Application Security Consortium [4.] i OWASP – Open Web Application Security Project [5.].Cilj obje organizacije je podizanje svijesti o sigurnosti Web aplikacija i

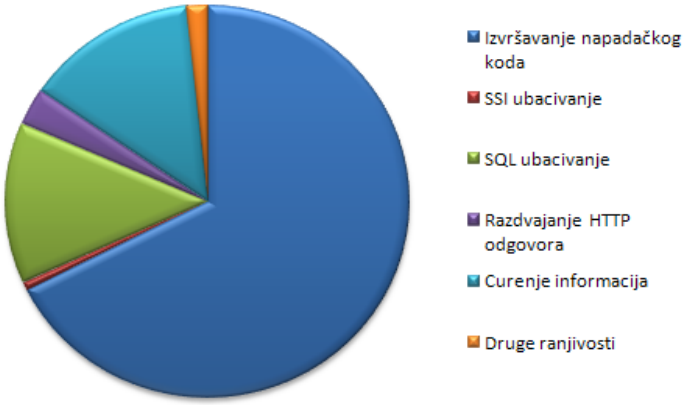
pružanje sigurnosne podrške korisnicima Web aplikacija. Svake godine, WASC i OWASP izdaju statističke pokazatelje pojavljivanja određenih vrsta ranjivosti Web aplikacija.

WASC je, koristeći automatizirane alate za otkrivanje ranjivosti, 2006. godine testirao 44,147 Web aplikacije i otkrio 148,029 ranjivosti. Udjeli određenih ranjivosti i njihova zastupljenost u Web aplikacijama su prikazani u tablici 2.2.

Tablica 2.2. Udio ranjivosti i njihova zastupljenost u Web aplikacijama (WASC)

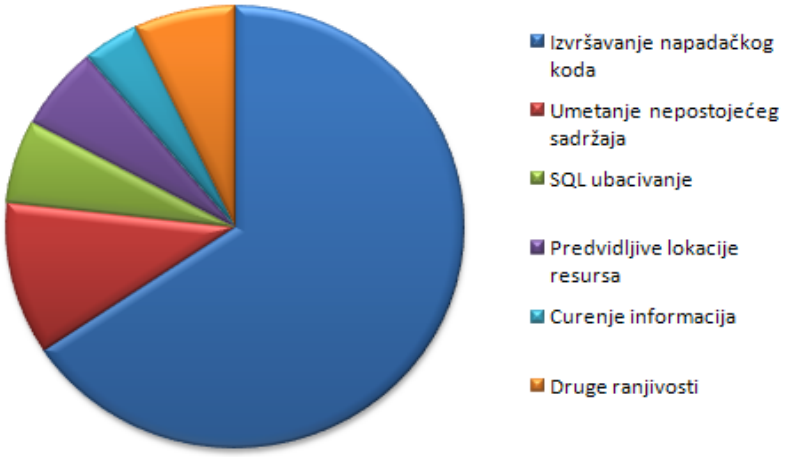
Klasa ranjivosti/napada	Broj pronađenih ranjivosti	Udio pronađenih ranjivosti	Broj ranjivih Web aplikacija	Udio ranjivih Web aplikacija
Gruba pretraga	66	0.04%	66	0.21%
Umetanje nepostojećeg sadržaja	663	0,045%	218	0.69%
Izvršavanje napadačkog koda	100,059	67,59%	26,531	84.57%
Indeksiranje mapa	292	0.20%	168	0.54%
Razdvajanje HTTP odgovora	4,487	3.03%	3,062	9.76%
Curenje informacija	20,518	13.86%	4,924	15,70%
Nedovoljna autentifikacija	84	0.06%	1	0.00%
Nedovoljna autorizacija	23	0.02%	4	0.01%
Nedovoljno trajanje sjednice	46	0.03%	1	0.00%
Izvršavanje naredbi OS-a	143	0.10%	44	0.14%
Korištenje prečaca	426	0.29%	374	1.19%
Previdljive lokacije resursa	651	0.44%	173	0.55%
SQL ubacivanje	19,607	13.25%	8.227	26.38%
SSI ubacivanje	950	0.64%	298	0.95%
Xpath ubacivanje	14	0.01%	6	0.02%
UKUPNO	**148,029**	**100.00%**	**44,147**	**100.00%**

Iz tablice je vidljivo da su najčešće ranjivosti: izvršavanje napadačkog koda, curenje informacija, SQL ubacivanje, razdvajanje HTTP odgovora i SSI ubacivanje. Na slici 2.1 su prikazane najzastupljenije ranjivosti za 2006. godinu.



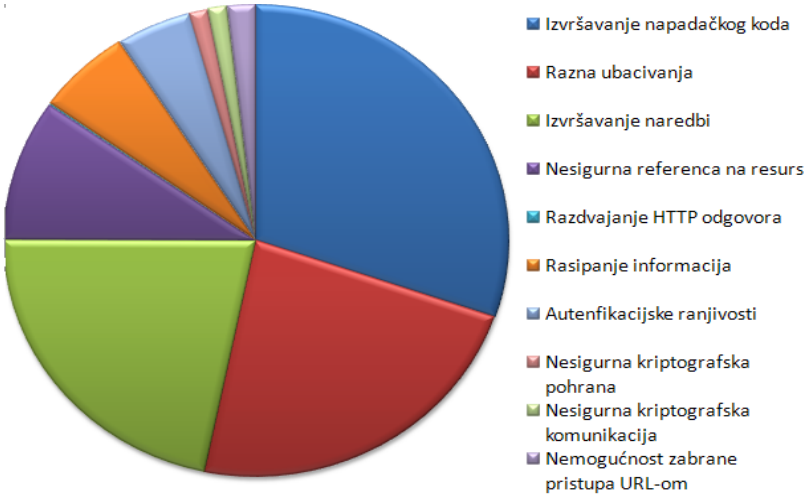
Slika 2.1. Najzastupljenije vrste ranjivosti Web aplikacija (WASC 2006)

Iako nepotpuni, najnoviji statistički podaci za 2007. prikazani na slici 2.2 pokazuju određenu promjenu u zastupljenosti ranjivosti. Vidljivo je da su potencijalni napadi sve više vezani uz ranjivosti korisničke strane. Tu se prvenstveno misli na izvršavanje napadačkog koda i umetanje nepostojećeg i lažnog sadržaja korisniku. Također se vidi povećanje rasipanja informacija, dok su se udjeli SQL i SSI ubacivanja smanjili.



Slika 2.2. Najzastupljenije vrste ranjivosti Web aplikacija (WASC početak 2007)

S druge strane, OWASP-ovi statistički pokazatelji udjela ranjivosti Web aplikacija su prikazani na slici 2.3.



Slika 2.3. Udjeli ranjivosti Web aplikacija (OWASP 2007)

Sudeći po statističkim pokazateljima obiju organizacija, dolazi se do zaključka da su određene ranjivosti zastupljenije i odskaču u udjelu od drugih vrsta ranjivosti. Tu se prvenstveno misli na izvršavanje napadačkog koda (XSS) i SQL ubacivanje.

U nastavku će se opisati većina spomenutih ranjivosti i napadačkih tehnika. Iako se opisi i klasifikacije većine ranjivosti podudaraju u obje organizacije, prednost će se dati WASC-ovim terminima i klasifikaciji, za razliku od rada navedenog u literaturi [6.] u kojem je dana prednost OWASP-ovoj klasifikaciji Klase ranjivosti, odnosno klase napada se dijele na:

- ranjivosti vezane uz autentifikaciju,
- ranjivosti vezane u autorizaciju,
- ranjivosti na klijentskoj strani,
- ranjivosti vezane uz izvršavanje naredbi,
- otkrivanje povjerljivih informacija i
- logičke ranjivosti.

2.1. Ranjivosti vezane uz autentifikaciju

Autentifikacija, s gledišta Web aplikacije, je proces kojim se provjerava identitet korisnika određene usluge Web aplikacije. Autentifikacija se provodi na temelju barem jednog od sljedeća tri mehanizma: "nečeg što imaš", "nečeg što znaš" i " onoga što jesi". Unutar ovog poglavlja objasnit će se napadačke tehnike kojima se zaobilazi i narušava autentifikacijski proces Web aplikacija.

2.1.1. Pretraživanje grubom silom

Pretraživanje grubom silom ili *Brute force* napadi je automatizirani proces koji se koristi metodom pogađanja i promašaja kako bi se otkrilo korisničko ime, lozinka, kriptografski ključ ili broj kreditne kartice. Ova vrsta napada je vrlo česta i relativno uspješna, uzimajući u obzir vrijeme trajanja napada koje može varirati od nekoliko minuta do nekoliko godina.

Mnoštvo današnjih sustava omogućava korisnicima korištenje jednostavnih lozinki ili kratkih kriptografskih ključeva. Korisnici najčešće odabiru lozinke koje su jednostavne za pamćenje i koje se često mogu naći u rječnicima. Kao posljedica takvog pristupa, potencijalni napadač može odabirom riječi iz rječnika stvoriti na tisuće netočnih upita kako bi otkrio važeću lozinku određenog korisnika. Ova vrsta napada postaje uspješna kada napadač pronađe lozinku određenog korisnika i pristupi njegovom korisničkom računu. Slična tehnika vrijedi i za sustave koji koriste kratke kriptografske ključeve; napadač umjesto korištenja riječi iz rječnika, koristi sve moguće vrijednosti ključa kako bi došao do točne vrijednosti korisničkog ključa.

U osnovi, postoje dvije vrste pretraživanja silom, normalno (uobičajni) i reverzno. Normalno pretraživanje koristi jedno korisničko ime i velik skup lozinki, dok reverzno pretraživanje koristi velik skup korisničkih imena i samo jednu lozinku. Reverzno pretraživanje se koristi u sustavima koji imaju na milijune korisničkih računa gdje su šanse da dva različita korisnika imaju istu lozinku velike.
Primjer alata za grubo pretraživanje je *Hydra,* paralelizirajući Web-cracker. *Hydra* posjeduje rječnik u kojeg se mogu upisati riječi koje se najčešće koriste od strane korisnika npr. Cola, Pepsi, Mercedes itd. Nakon što se specificira sadržaj rječnika, *Hydra-i* se zadaje put do tražene aplikacije, te započinje pretraživanje silom za traženo korisničko ime. Ishod isključivo ovisi o veličini i sadržaju rječnika.

2.1.2. Nedovoljna razina autentifikacije

Nedovoljna razina autentifikacije nastupa kada Web aplikacija omogućava potencijalnom napadaču pristup osjetljivom sadržaju ili funkcionalnosti, a da se pritom napadač nije propisno autentificirao. Ograničavanje pristupa osjetljivom sadržaju je uobičajno za većinu Web aplikacija, dok je pristup funkcionalnosti uobičajan uglavnom za administratorske Web alate.

Kako bi se Web aplikacija zaštitila procesom autentifikacije, njeni resursi se najčešće štite skrivanjem lokacija-URL-a. Iako potencijalni napadač ne zna o kojima se točno resursima radi, on njima može direktno pristupiti korištenjem URL-a. Određeni URL se može otkriti korištenjem alata za grubo pretraživanje kojima se pronalaze lokacije mapa i datoteka,

poruke s greškama i administratorski zapisi (engl. *logs*).Ove resurse je potrebno dodatno zaštititi dozvolama ili drugim metodama, kako se ne bi zloupotrijebili.

Primjer nedovoljne razine autentifikacije su Web aplikacije koje posjeduju administratorsku mapu */admin/* direktno unutar osnovne mape Web aplikacije (*root*). Iako ova mapa nije povezana (nema link) ni s jednim dokumentom Web aplikacije koji se daje na raspolaganje korisniku, korisnik može direktno pristupiti mapi korištenjem Web preglednika i na taj način zaobići proces autentifikacije. Najčešći uzrok nedovoljne razine autentifikacije je administratorsko mišljenje da pristup administratorskoj mapi nije moguć ukoliko ne postoji link na /*admin/* mapu, pa uopće ne smatra potrebnim postavljanje dozvola pristupa na /*admin/* mapu.

2.1.3. Nedovoljna zaštita korisnikove lozinke

Nedovoljnom zaštitom korisnikove lozinke se omogućava napadaču ilegalno pribavljanje, promjenu ili obnovu lozinke drugog korisnika.

Proces autentifikacije određene Web aplikacije zahtjeva od korisnika pamćenje lozinke ili određene fraze koja omogućava pristup korisničkom računu. Korisnik bi trebao biti jedina osoba koja točno zna lozinku. Kako vrijeme prolazi, korisnikova sposobnost pamćenja lozinke se smanjuje, pogotovo ako se uzme u obzir da korisnik ima desetak i više korisničkih računa za različite Web aplikacije. Ukoliko korisnik zaboravi lozinku, tada pristupa procesu za obnovu lozinke. Najčešći primjer procesa za obnovu lozinke je proces koji koristi princip "tajnog pitanja" (engl. *Secret question*) koje korisnik definira prilikom registracije računa. Ukoliko korisnik zaboravi lozinku, odgovorom na "tajno pitanje" može obnoviti svoju lozinku. Još jedan primjer procesa za obnovu lozinke je specificiranje trika (engl. *hint*) koji pomaže korisniku da se prisjeti svoje lozinke. Drugi mehanizmi obnove zahtijevaju od korisnika osobne podatke kao što su adresa elektronske pošte, adresa prebivališta, broj kartice i slično, kako bi potvrdili svoj identitet.

Najveći problem procesa za obnovu je mogućnost prevare procesa od strane napadača. To se najčešće događa kada su informacije potrebne za provjeru korisničkog identiteta lako dostupne ili se daju naslutiti. Sustavi za obnovu se najčešće kompromitiraju korištenje alata za grubo pretraživanje, naslijeđenom ranjivošću sustava ili naslutljivim "tajnim pitanjima".

Primjeri napada:

- Napadač pošalje zahtjev s određenim korisničkim imenom procesu za obnovu lozinke. Proces odgovori na njegov zahtjev s "tajnim pitanjem" tipa "U kojem mjestu si rođen?". Ukoliko napadač posjeduje alat za grubo pretraživanje u čijem se rječniku zapisani svi gradovi u Hrvatskoj, postoji velika mogućnost da će ponuditi točan odgovor, odnosno prevariti proces za obnovu i saznati korisnikovu lozinku.
- Korisnik koristi trik kako bi upamtio svoju lozinku tipa "Ime+datum rođenja". Ukoliko napadač sazna o kojem se triku radi (trik je najčešće dostupan) može značajno smanjiti područje pretrage, odnosno smanjiti broj zapisa u rječniku alata za grubo pretraživanje i povećati mogućnost uspješnog napada.

2.2. Ranjivosti vezane uz autorizaciju

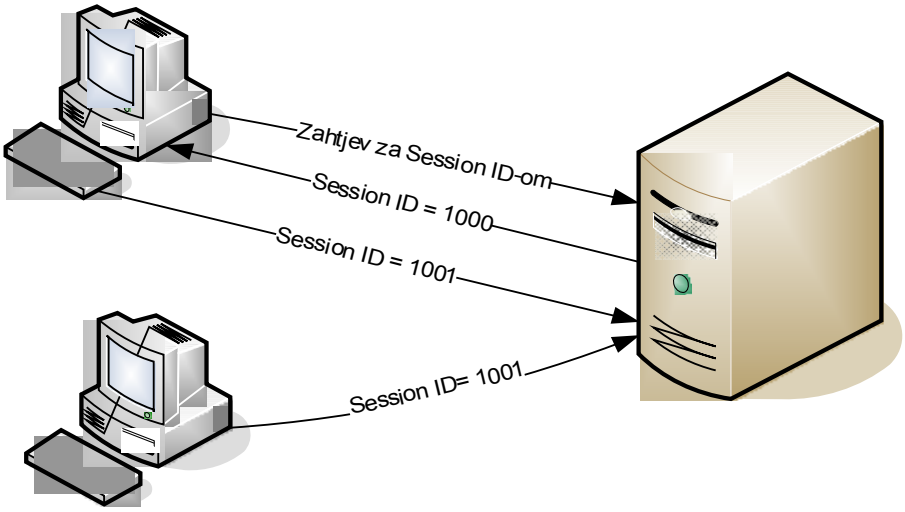
Autorizacija, s gledišta Web aplikacija, je proces kojim se utvrđuje da li određeni korisnik, usluga ili aplikacija ima potrebna dopuštenja za izvođenje određene radnje. Većina Web aplikacija određenom korisniku daje na raspolaganje točno određene sadržaje i funkcionalnosti zaštićene određenom razinom ovlasti, dok su neki drugi sadržaji i funkcionalnosti zabranjeni istom korisniku. Unutar ovog poglavlja opisati će se napadačke tehnike koje omogućavaju napadaču povećanje njihovih privilegija i pristup zabranjenim resursima.

2.2.1. Nagađanje sjedničkog identifikacijskog broja

Web aplikacije najčešće za izmjenu informacija s korisnicima koriste HTTP koji spada u grupu protokola bez stanja (engl. *stateless*). To znači da će korisnik Web aplikacije prilikom svakog posjeta biti promatran od strane Web poslužitelja kao da je prvi put pristupio aplikaciji. Drugim riječima, poslužitelj zaboravlja sve o korisniku nakon korisnikovog zahtjeva, osim ako na neki način ne označi korisnika. To označavanje se provodi dodjeljivanjem vjerodajnog identifikacijskog broja poznatog kao identifikator sjednice (engl. *Session ID*). U današnjim Web preglednicima, tehnološki sinonim za identifikator sjednice je kolačić (engl. *Cookie*). Identifikator sjednice je znakovni niz koji se čuva u memoriji Web preglednika, a postavlja se od strane Web aplikacije. *Identifikator sjednice* ima svoj životni vijek kojim je točno određeno koliko vremena određeni korisnik može koristiti usluge Web aplikacije, a da se od njega ne zahtijevaju identifikacijski podaci kojim bi se produžila valjanost sjedničkog identifikatora .

Nagađanje sjedničkog identifikacijskog broja je napadačka tehnika kojom se otima identitet drugog korisnika. Napadačeva namjera je saznati identifikator sjednice koji je dodijeljen određenom korisniku od strane određene Web aplikacije kako bi iskoristio korisnikove privilegije za kompromitirajuće radnje. Mnoštvo Web aplikacija je dizajnirano tako da bi se mogla provesti autentifikacigućilo praćenje rada korisnika nakon uspostave veze korisnik-Web aplikacija. Da bi se ovo ostvarilo, korisnici moraju dokazati svoj identitet Web aplikaciji, najčešće upisivanjem vlastitog korisničkog imena i lozinke. Kako bi se spriječilo unošenje povjerljivih korisničkih podataka tokom svake transakcije, Web aplikacije generiraju jedinstveni identifikator sjednice pomoću kojeg se korisnik autentificira. Ako je napadač sposoban pretpostaviti ili naslutiti identifikator sjednice određenog korisnika, tada je moguća zloupotreba korisničkih privilegija i povreda autorizacije.

Primjer za ovu vrstu napada su Web aplikacije koje generiraju identifikator sjednice koristeći predvidljive i jednostavne algoritme, pogotovo u slučajevima gdje se trenutni identifikator generira jednostavno inkrementirajući prethodno generirani identifikator. Sjednički identifikator se najčešće pohranjuje unutar datoteke ili URL-a, a također ga je moguće pohraniti unutar skrivenog polja (engl. *Hidden form*). Ako napadač odredi algoritam pomoću kojeg se generiraju identifikatori sjednice, tada se napad može izvesti na način kao što je prikazano na slici 2.4:



Slika 2.4. Nagađanje sjedničkog identifikatora

Slijed napada je sljedeći:

1) Napadač se spaja na Web aplikaciju kako bi dobio trenutni sjednički identifikator. Web aplikacija mu vraća sjednički identifikator s vrijednošću 1000.
2) Pošto napadač zna da se radi o inkrementirajućem algoritmu za generiranje sjedničkog identifikatora, napadač jednostavno izračuna vrijednost sljedećeg identifikatora (vrijednost 1001).
3) Napadač izmjeni vrijednost znakovnog niza unutar sjedničkog identifikatora ili URL-a na vrijednost 1001 i upućuje zahtjeve Web aplikaciji sve dok se ne prijavi sljedeći korisnik. Nakon što se korisnik prijavi, napadač može koristiti njegove privilegije koristeći izračunati sjednički identifikator.

2.2.2. Nedovoljna autorizacija

Nedovoljnom autorizacijom, Web aplikacija omogućava napadaču pristup sadržaju ili funkcionalnosti koja bi inače trebala biti zaštićena višom razinom sigurnosti.

Provođenjem autentifikacijskog procesa, korisniku Web aplikacije se ne daju prava za potpuni pristup cjelokupnom sadržaju i funkcionalnosti. Autorizacijski proces se provodi poslije procesa autentifikacije prisiljavajući korisnike, usluge ili aplikacije na ograničeno korištenje resursa. Autorizacijska prava se najčešće uređuju sigurnosnom politikom. U kontekstu Web aplikacija, resursi koje treba dodatno zaštiti su podaci isključivo namijenjeni administratoru Web aplikacije.

Primjer nedovoljne autorizacije su aplikacije koje su administratorske podatke skrivali u mapama s nazivima */admin/* ili */logs/*, kojima je mogao pristupiti bilo koji autentificirani korisnik i poduzeti neželjene radnje kao što su rekonfiguracija poslužitelja itd.

2.2.3. Nedovoljna kontrola trajanja korištenja usluge

Nedovoljnom kontrolom trajanja korištenja usluge, Web aplikacija omogućava napadaču korištenje starih sjedničkih identifikatora (engl. *Session ID)* za autorizaciju.

Kao što je već rečeno, Web aplikacije se služe HTTP *stateless* protokolom i zbog toga koriste identifikatore sjednice kako bi identificirali korisnike od zahtjeva do zahtjeva. Kao posljedica, svaki identifikator mora biti tajno sačuvan kako bi se spriječio višekorisnički pristup istom korisničkom računu. Nedostatak kontrole trajanja usluge, odnosno nedostatak kontrole životnog vijeka sjedničkog identifikatora može povećati rizik od određenih vrsta napada. Na primjer, napadač može prisluškivati mrežu i preuzimati pakete koji sadrže identifikatore, ili može koristi napad vezan uz izvršavanje napadačkog koda (XSS) kako bi došao u posjed identifikatora. Iako skraćivanje životnog vijeka identifikatora ne može spriječiti upotrebu identifikatora koji je nedavno ukraden, ono će spriječiti uzastopno i trajno korištenje pribavljenog identifikatora. S druge strane, dugo trajanje sjedničkog identifikatora povećava napadačevu šansu da sazna ili izračuna ispravni korisnički sjednički identifikator, ali smanjuje broj potrebnih korisničkih prijava na korisnički račun određene Web aplikacije.

Primjer nedovoljne kontrole trajanja usluge je korištenje višekorisničke (engl. *Shared*) računalne okoline (Internet caffe, knjižnica itd.). Ukoliko se korisnik prikladno ne odjavi s vlastitog korisničkog računa, tada postoji mogućnost da će sljedeći korisnik računala upotrebom gumba za povratak (engl. *back button*) *Web* preglednika, pregledati sve stranice posjećene od strane prvog korisnika (žrtve).Nadalje, ako prvi korisnik odjavom nije prekinio sjednicu, tada je moguće da drugi korisnik pregledavajući povijest (engl. *History*) posjećenih stranica dođe do određenih sadržaja za koje mu inače trebaju prava prvog korisnika.

2.2.4. Fiksacija usluge (sjednice)

Fiksacijom usluge napadač podvaljuje korisniku fiksni identifikator sjednice (engl. *Session ID)*, koji kasnije napadač koristi za autorizaciju.

Postoji mnoštvo napadačkih tehnika pomoću kojih se identifikator sjednice može fiksirati na neku eksplicitnu vrijednost. Najčešće tehnike su korištenje XSS napada i posebno prilagođeni HTTP zahtjevi. Cilj ove tehnike je, nakon što napadač postavi identifikator sjednice na neku fiksnu vrijednost, pričeka korisnika da se prijavi na tu predefiniranu sjednicu s podvaljenim identifikatorom. Nakon što se korisnik prijavi, napadač upotrebljava taj identifikator sjednice, pridobiva korisnikov identitet i koristi njegove privilegije.

Bez aktivne zaštite protiv fiksacije usluge, napad se može izvesti nad bilo kojom Web aplikacijom koja koristi sjedničke identifikatore kako bi identificirala korisnike. Web aplikacije najčešće koriste identifikatore sjednice, nešto manje URL i skrivena polja za prijenos informacija o sjednici. Nažalost, upravo Web aplikacije koje koriste sjedničke identifikatore su najranjivije jer je većina današnjih napada usmjerena baš prema kompromitaciji sjedničkih identifikatora.

Uobičajna fiksacija usluge se sastoji od 3 sljedeće faze:

1. Postavljanje sjedničkog identifikatora

   Napadač postavlja klopku za određenu Web aplikaciju kako bi pridobio njen sjednički identifikator ili postavlja neki vlastito skrojeni sjednički identifikator (ovisi o sustavu na koji aplikacija radi).

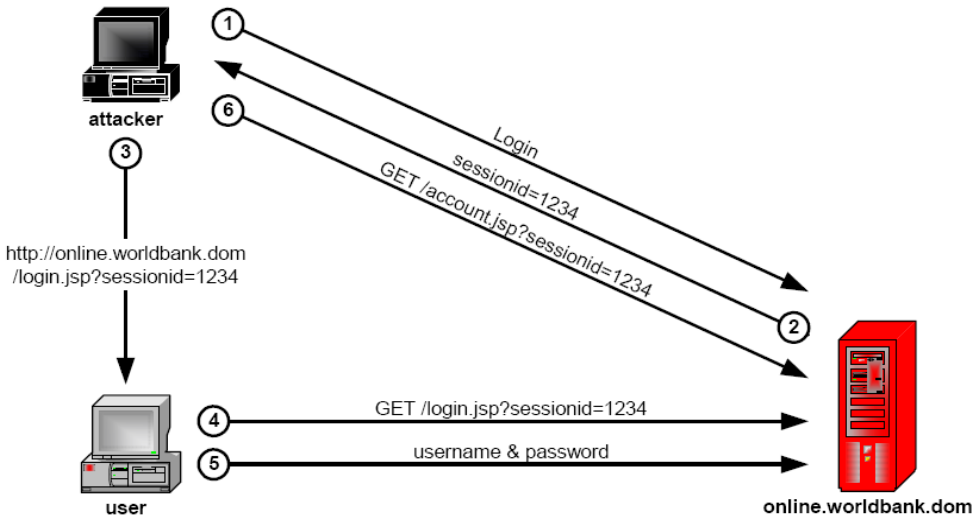
2. Fiksacija usluge

Napadač predstavlja klopku ili vlastito skrojeni sjednički identifikator određenom korisniku, te ga uvjerava da taj identifikator potječe od same Web aplikacije.

3. Ulazak i korištenje usluge

   Napadač čeka dok se korisnik ne prijavi na traženu Web aplikaciju. Kada to korisnik napravi, tada napadač posjeduje fiksirani sjednički identifikator kojim preuzima korisnička prava i identitet

Pogledajmo sljedeći primjer koji prikazuje fiksaciju usluge. Slika 2.5 prikazuje Web poslužitelj *online.worldbank.com* na kojem je postavljena bankovna Web aplikacija usmjerena prema korisnicima. Sjednički identifikatori se prosljeđuju korisnicima koristeći URL argument *sessionid.*



Slika 2.5. Fiksacija usluge

U prvom koraku, napadač koji je također legitimni korisnik bankovnog sustava, se ulogira na poslužioca (1) i dobije *session ID* s vrijednošću 1234 (2). Nakon toga napadač pošalje posebno skrojeni link `http://online.worldbank.dom/login.jsp?sessionid=1234` korisniku, te pokušava namamiti korisnika da pritisne link (3). Korisnik pritišće link, koji otvara poslužiteljsku stranicu za prijavu u korisnikovom Web pregledniku (4). Potrebno je naglasiti da je za prilikom prihvata zahtjeva `login.jsp?sessionid=1234`, Web aplikacija već otvorila sjednicu za danog korisnika i nova sjednica nije potrebna. Na kraju, korisnik upisuje svoje korisničko ime i lozinku (5) te ih prosljeđuje poslužitelju kako bi mu dodijelio pristup bankovnom računu. Ali, napadač također zna sjednički identifikator, te može pristupiti korisnikovom računu koristeći `account.jsp?sessionid=1234` (6). Kako je sjednički identifikator već unaprijed fiksiran prije nego što se korisnik prijavio na sustav, kažemo da se korisnik prijavio u sjednicu postavljenu od napadača.

2.3. Ranjivosti klijentske strane

Napadi na klijentsku stranu su orijentirani prema korisnicima Web aplikacija. Kada korisnik posjeti određenu Web aplikaciju, uspostavlja se povjerenje između korisnika i poslužitelja, odnosno Web aplikacije. Korisnik očekuje od Web aplikacije dostavljanje ispravnog sadržaj i da se prilikom njegovog korištenja Web aplikacije neće dogoditi nikakav napad na njega. Mnoštvo napadačkih tehnika može ugroziti ovaj odnos, ali zbog opširnosti će se opisati samo dvije najbitnije tehnike: ubacivanje nepostojećeg sadržaja (engl. *Content spoofing*) i izvršavanje napadačkog koda (engl. *Cross-site scripting*) u korisnikovom Web pregledniku.

2.3.1. Ubacivanje nepostojećeg sadržaja

Ubacivanje nepostojećeg sadržaja je vrsta napada kojom napadač želi uvjeriti korisnika da je određeni sadržaj legitiman i da ne potječe s nekog vanjskog izvora. Ova napadačka tehnika je najčešće usmjerena prema Web aplikacijama koje dinamički generiraju URL-ove prema svom HTML sadržaju.

Primjerice, neka korisnik želi pristupiti nekoj Web aplikaciji putem sljedećeg URL-a: `http://www.fer.hr/page?frame_src=http://www.fer.hr/file.html`. Napadač može izmjenom sadržaja HTML dokumenta promijeniti izvor *frame_src* parametra u `frame_src=http://napad.hr/file.html` i umetnuti svoj sadržaj u `file.html`, te proslijediti skrojeni link korisniku bilo putem elektroničke pošte, IM-a (engl. *instant messanger*) ili poslati link na forum. Korisnik posjećuje ovu stranicu, vidi domenu `http://www.fer.hr` u Web pregledniku i uvjeren je da je sadržaj koji promatra autentičan i s izvorne lokacije, a u biti sadržaj potječe s napadačevog izvora `http://napad.hr/file.html`.

Iz ovog primjera je vidljivo da je ugrožen povjerljiv odnos između korisnika i Web aplikacije. Ova napadačka tehnika se najčešće koristiti za stvaranje lažnih Web stranica za prijavu korisnika, gdje se na relativno jednostavan način može ukrasti korisnikov identitet.

2.3.2. Izvršavane napadačkog koda

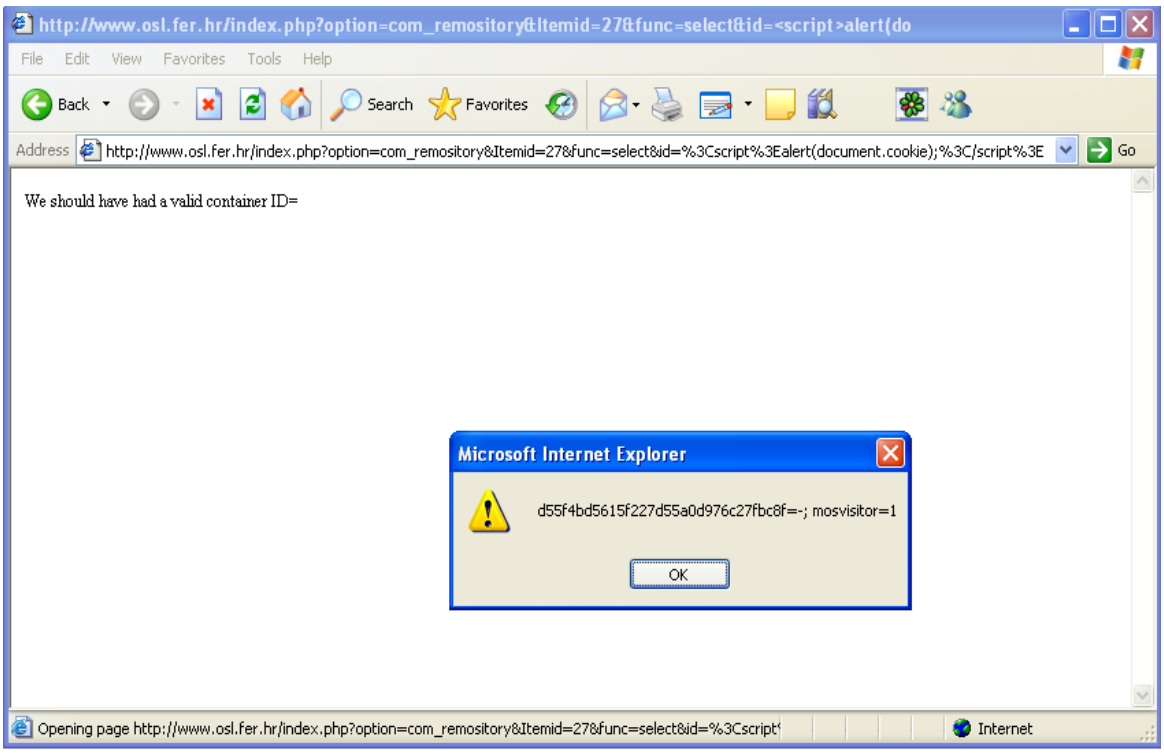
Izvršavane napadačkog koda ili *cross-site scripting* (XSS) je napadačka tehnika koja prisiljava Web aplikaciju na proslijeđivanje napadačkog izvršnog koda korisniku, koji se zatim učitava u korisnikovom Web pregledniku i izvršava. Napadački kod je najčešće napisan korištenjem JavaScript skriptnog jezika, ali također i drugih programskih jezika koju su podržani od strane korisničkog Web preglednika : VBScript, ActiveX, Java i Flash.

Kada napadač uspije potaknuti korisnički Web preglednik na izvršavanje napadačkog koda, kod će se izvršavati unutar sigurnosne zone pristupajuće Web aplikacije. Koristeći ovu privilegiju, napadački kod će moći čitati, promijeniti ili proslijediti osjetljive podatke koji su dani na raspolaganje Web pregledniku. Na taj način, ova napadačka tehnika se može iskoristiti za krađu korisničkih računa (krađa sjedničkog identifikatora), usmjeravanje Web preglednika na neke druge lokacije, ili prosljeđivanje štetnog sadržaja od strane Web aplikacije. Stoga, *XSS* napadi ugrožavaju povjerljivi odnos između korisnika i Web aplikacije.

Općenito, postoje dvije vrste XSS napada, neustrajni (engl. *non-persistent*) i ustrajni (engl. *Persistent*). Neustrajni napadi navode korisnika na posjećivanje posebno skrojenih linkova

koji su povezani s štetnim kodom. Kada korisnik posjeti link, kod koji je pohranjen unutar URL-a će se izvršiti unutar korisničkog Web preglednika. Oprezniji korisnici mogu uviditi opasnost ako unutar sadržaja linka primijete skriptu, pa napadači najčešće konvertiraju kod koristeći *Hex* kodiranje kako bi prikrili trag skripti i zavarali korisnika. S druge strane, ustrajni napadi se događaju kada je određeni štetni kod pohranjen unutar Web aplikaciji neko određeno vrijeme. Te aplikacije su najčešće portali, *Web-mail* ili *Web-chat* aplikacije. Pritom nije nužno da korisnik pritisne neki link, već je dovoljno da jednostavno pregleda sadržaj Web stranice koja sadrži štetni kod.

Primjerice, ispituje se postojanje *XSS* ranjivosti na službenoj stranici FER OSL-a. Općenito, da bi ispitali pojedinu Web aplikaciju na *XSS ranjivost*, potrebno je ubaciti jednostavnu Javascript *alert* poruku (*<script>alert(XSS)</script>*) u dio Web aplikacije koji se može vidjeti (najčešće URL ili poruke unutar nekog foruma). Nakon što se postavi poruka, potrebno je pogledati da li će Web preglednik reagirati *pop-up* prozorom s porukom . Ako se pojavi prozor kao na slici 2.6, to je znak da se skripta izvršila i da postoji *XSS* ranjivost.



Slika 2.6. XSS ranjivost

U ovom slučaju, posljedica *XSS* napada je krađa sjedničkog identifikatora. Koristeći sljedeći skrojeni link koji u sebi sadrži skriptu prikaže se *pop-up* prozor koji ispiše korisnički sjednički identifikator*:*

```
http://www.osl.fer.hr/index.php?option=com_remository&Itemid=27
&func=select&id=<script>alert(document.cookie);</script>
```

Iz primjera je vidljiv način na koji se može ukrasti korisnički identitet.

Ovo je jedna od češćih napadačkih tehnika, pa je potrebno provesti dodatne sigurnosne mjere kako bi se se izbjegla ova vrsta napada:

- Onemogućavanje skripti kada one nisu potrebne. Na ovaj način se sprečava izvršavanje koda unutar korisničkog Web preglednika putem skripti (unutar URL-a), ali i dalje postoji opasnost od posebno skrojenih napadačkih HTML dokumenata koji se najčešće prosljeđuju korisniku putem elektroničke pošte.
- Filtriranje korisničkih zahtjeva. Na ovaj način, Web aplikacija uvijek provjerava korisničke zahtjeve i filtrira posebne meta znakove definirane HTML specifikacijom kako bi ustanovila da li korisnički zahtjev sadrži skriptu. Ukoliko zahtjev sadrži skriptu, Web aplikacija sprečava prikazivanje štetnog HTML dokumenta unutar korisničkog Web preglednika.
- Kodiranje stranica. XSS napadi se mogu izbjeći ukoliko Web poslužitelj pravilno kodira generirane stranice kako bi onemogućio izvršavanje skripti.

2.4. Ranjivosti vezane uz izvršavanje naredbi

Ranjivosti, odnosno napadi vezani uz izvršavanje naredbi podrazumijevaju napade koji su posebno dizajnirani kako bi izvršili kompromitirajuće naredbe nad Web aplikacijom. Sve Web aplikacije procesiraju korisnički unesene ulazne podatke kako bi formirale i ispunile zahtjeve. Tu je javlja problem jer se korisnički podaci često koriste za stvaranje posebnih naredbi unutar dinamičkog sadržaja Web stranice. Ako se korisnički podaci loše ukomponiraju unutar sadržaja Web aplikacije, tada potencijalni napadač može izmijeniti ili potaknuti izvođenje pojedinih naredbi.

2.4.1. Prelijevanje spremnika

Prelijevanje spremnika je ranjivost koja dopušta promjenu ili onemogućavanje toka izvođenja aplikacijskih procesa na način da se prebrišu određene dijelovi memorije. Prelijevanje spremnika je u biti obično programsko ugrožavanje memorijskog spremnika koje rezultira pojavom greške unutar programa, odnosno Web aplikacije. Ova greška nastupa kada podatak upisan u memoriju pređe alociranu veličinu memorije spremnika. Pošto je spremnik preliven, susjedne memorijske adrese su prebrisane i uzrokuju kvar programa. Ukoliko se napravi propust prilikom dizajniranja Web aplikacije, tada postoji mogućnost da se posebno skrojenim ulaznim podacima izazove prelijevanje memorijskog spremnika i naruši sigurnost Web aplikacije.

Prelijevanje spremnika se najčešće koristi kao DoS napad koji uzrokuje kvar i rušenje Web aplikacije. Također, napad može promijeniti tok izvođenja aplikacije i uzrokovati neželjene akcije. Ovo se može realizirati ukoliko se promijene adrese pokazivača stoga ili vrijednosti programskih varijabli, odnosno promjeni tok izvođenja programa kako bi se izvršile različite štetne naredbe.

Ova napadačka tehnika se učestalo koristila za rušenje Web poslužitelja. Na svu sreću, vrlo rijetko ugrožava Web aplikacije. Razlog tomu je što napadač mora detaljno analizirati izvorni kod aplikacije kako bi otkrio potencijalnu ranjivost, pa se većina napada uglavnom svodi na “slijepe” napade koji su rijetko uspješni. Prelijevanje spremnika se najčešće javlja u CGI aplikacijama pisanim u C ili C++ programskom jeziku.

2.4.2. Napadi formatiranim znakovnim nizovima

*N*apadi formatiranim znakovnim nizovima mijenjaju programski tok aplikacije koristeći različite oblike formatirajućih naredbi kako bi dobili pristup određenom memorijskom prostoru. Ova vrsta ranjivosti najčešće nastaje zbog lijenosti programera koji prilikom pisanja izvornog koda aplikacije ne koriste ispravne oblike formatirajućih naredbi.

Primjer jednog programerskog propusta je korištenje formata naredbe `printf(string)` umjesto `printf("%s", string)`. U ovom primjeru programer je prvom naredbom htio jednostavno ispisati određeni znakovni niz `string`, a pritom nije razmišljao o načinu na koji će se taj znakovni niz interpretirati. Naime, niz će se interpretirati kao formatirajući znakovni niz, odnosno pretraživati će se posebni formatirajući znakovi poput `"%d"` pomoću kojih se može dobiti informacija o vrijednostima stoga programa. U biti, programer ovim propustom omogućava napadaču pogled u memoriju na temelju kojih napadač dobiva dovoljno informacija za dodatne napade. Analogno ovom primjeru, napadač može direktno unositi formatirane nizove koji sadrže specijalne znakove za određene C/C++ funkcije (npr. `fprintf, printf, sprintf, syslog, ...`).

Napadi formatiranim znakovnim nizovima se koriste za:

- Čitanje podataka s stoga, pri čemu napadač koristi konverzijski znak `"%x"` unutar `printf` naredbe te kao rezultat dobiva vrijednosti stoga.
- Čitanje znakovnih nizova. Ukoliko se rezultat `printf` naredbe vrati napadaču, tada on koristeći konverzijski znak `"%s"` može čitati znakovne nizove iz memorije.
- Pisanje cjelobrojnih konstanti u procesnu memoriju. Ukoliko napadač koristi `"%n"` konverzijski znak, tada napadač može upisati cjelobrojnu konstatnu na bilo koju lokaciju u memoriji. Primjerice, može izmjeniti kontrolne zastavice unutar programa kako bi promijenio tok izvođenja programa.

2.4.3. Izvršavanje OS naredbi

Izvršavanje OS naredbi je napadačka tehnika koja se koristi manipulacijom aplikacijskih ulaznih podataka kako bi se izvršile naredbe operacijskog sustava na kojem je pokrenuta Web aplikacija.

Kada Web aplikacija nedovoljno provjerava smisao korisnički unesenih podataka, tada postoji mogućnost prevare aplikacije i izvršavanja određene naredbe operacijskog sustava. Pri čemu se izvršene naredbe pokreću s dozvolama komponenti koje su pokrenule njeno izvođene (npr poslužitelj baze podataka, Web poslužitelj).

Primjer je Web aplikacija kojoj napadač putem URL-a proslijedi sljedeći podatak: `http://primjer/temp.php?dir=%3Bcat%20/etc/passwd`. Zbog propusta aplikacije da sanira vrijednost varijable `dir`, napadač će ovom naredbom doći do sadržaja `/etc/passwd` datoteke koja sadrži povjerljive podatke.Koristeći ovu napadačku tehniku, napadač čak može koristeći TFTP protokol ubaciti svoje alate i u potpunosti preuzeti kontrolu nad sustavom.

2.4.4. SQL ubacivanje

SQL ubacivanje je napadačka tehnika koja se koristi kako bi se ugrozila sigurnost Web aplikacije koja konstruira SQL izjave iz korisnički unesenih podataka.*Structured Query Language* (SQL) je specijalizirani programski jezik koji se koristi za rukovanje bazama podataka putem izjava i upita. SQL je ANSI i ISO standard, stoga je u današnje vrijeme najraširenjiji jezik za rukovanje bazama podataka koji koristi većina današnjih Web aplikacija. Web aplikacije mogu korisiti korisnički unesene podatke kako bi stvorile posebne SQL izjave za rad s dinamički generiranim Web stranicama.

Kada Web aplikacije ne uspiju pravilno sanirati korisnički unesene podatke, tada postoji mogućnost izmjene konstrukcije pozadinske SQL izjave od strane napadača. Ako napadač uspije izmjeniti SQL izjavu, proces će se pokrenuti s dozvolama komponente koja je pokrenula naredbu (npr poslužitelj baze podataka, Web poslužitelj). Posljedica ove napadačke tehnike je preuzimanje kontrole nad bazom podataka i čak izvršavanje naredbi nad sustavom.

Primjer je Web aplikacija za autentifikaciju koja sadrži sljedeći kod za konstrukciju SQL izjava:

```
String SQLQuery = "SELECT Username FROM Users WHERE Username = '" +
                  username + "' AND Password = '" + password + "'";
```

U ovom kodu, programer uzima korisnički unesene podatke iz autentifikacijskog obrasca i direktno ih ubacuje u SQL izjavu (putem varijabli `username` i `password`). Pretpostavka je da napadač u autentifikacijski obrazac za korisničko ime i lozinku upisuje sljedeće:

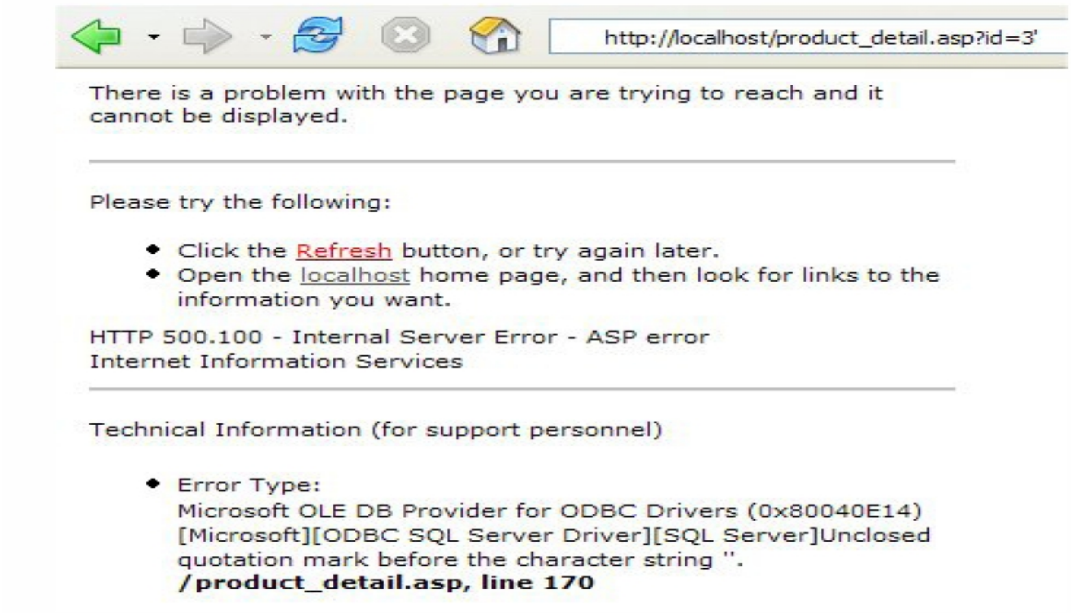
```
Korisničko ime:  ' OR ''='
Lozinka: ' OR ''='
```

Tada će rezultirajuća SQL izjava izgledati ovako:

```
SELECT Username FROM Users WHERE Username = '' OR''=''
                              AND Password = '' OR ''=''
```

Vidljivo je, da će se umjesto uspoređivanja korisnički unesenih podataka s podacima unutar korisničke tablice (`Users`), uspoređivati '' (prazan niz) s '' (prazan niz). Stoga će rezultat ove SQL izjave uvijek biti istinit i napadač će se uspjeti prijaviti na sustav kao prvi korisnik u korisničkoj (`Users`) tablici.

Najefektivnija metoda za lociranje SQL ranjivosti je ručno pretraživanje – proučavanje različitih aplikacijskih ulaznih podataka i ubacivanje posebnih znakova. Kako se pri radu se bazama najčešće dobije povratna informacija u obliku stranice s porukom o pogrešci, potencijalni napadač može približno odrediti sintaksu SQL izjava nad određenom bazom podataka i izvesti *SQL ubacivanje*. Stoga, pri razvoju Web aplikacije posebnu pozornost treba obratiti na detaljnost stranica s porukom o grešci kako se ne bi otkrile suvišne informacije kao u na slici 2.7.



Slika 2.7. Prikaz stranice s porukom greške

Iz slike je vidljivo da su u stranici s greškom ispisane dodatne informacije o bazi podataka : vrsta poslužitelja baze podataka , vrsta pogreške te čak linija koda u kojoj se pogreška dogodila. Na ovaj način se napadaču malo pomalo otkriva struktura same baze podataka.

Općenito, postoje dvije vrste SQL ubacivanja : slijepo i normalno SQL ubacivanje. Slijepo SQL ubacivanje se već spomenulo u prijašnjem odlomku kad se spominjalo ručno pretraživanje. Dakle, prilikom slijepog SQL ubacivanja, umjesto vraćanja jednostavne i kratke poruke o pogrešci, poslužitelj vraća detaljan opis o pogrešci te tako pomogne napadaču. Na temelju te poruke pokušava se realizirati SQL ubacivanje postavljajući istinitu i lažnu izjavu kao vrijednost određenog parametra. Slijepo ubacivanje je prikazan sljedećim primjerom:

`http://example/article.asp?ID=2+and+1=1` istinita izjava
`http://example/article.asp?ID=2+and+1=0` lažna izjava

Normalno SQL ubacivanje koristi `union select` izjavu kao vrijednost parametra kako bi napadač utvrdio da li može pristupiti bazi podataka, kao što prikazuje sljedeći primjer:

```
http://example/article.asp?ID=2+union+all+select+name+from+sysobjects
```

Uspjeh ovog napada ovisi o broju stupaca u traženoj tablici, pa je na napadaču da odredi ispravan broj stupaca (atributa). Ako to uspije odrediti, dobiti će uvid u bazu podataka.

2.4.5. LDAP i Xpath ubacivanje

LDAP i *XPath ubacivanje* su po smislu i načinu izvedbe identični *SQL ubacivanju* . I dalje je kritično saniranje korisnički unesenih podataka koji se prenose u *LDAP* i *XPath* izjave. Stoga će se u ovom dijelu samo opisati svrhe *LDAP-a* i *XPath-a*.

Lightweight Directory Access Protocol (*LDAP*) je otvoreni standard za postavljanje upita i manipulaciju X.500 direktorijima (mapama). *LDAP* se temelji na TCP-u, a Web aplikacije iz korisnički unesenim podataka kreiraju *LDAP* izjave za rad s dinamičkim Web stranicama.

XPath je jezik koji se koristi kako bi se referencirali određeni dijelovi XML dokument. Može se koristiti direktno od strane Web aplikacije kako bi se postavili upiti na da XML dokumentom. *Xpath* izjave su po strukturi vrlo slične SQL izjavama, te omogućavaju izdvajanje pojedinih elemenata i atributa iz XML dokumenta.

2.5. Otkrivanje povjerljivih informacija

Otkrivanjem povjerljivih informacija, potencijalni napadač namjerava dobiti određene sistemske informacije o Web aplikaciji. Sistemske informacije uključuju distribuciju programske podrške, verziju ili razine zakrpa. Također, informacije mogu sadržavati lokacije privremenih (*temp*) ili *backup* datoteka. Mnoštvo današnjih Web aplikacija otkriva određenu količinu podataka, ali je najbolje sakriti što veću količinu podataka kako bi se smanjili rizici napada i povećala sigurnost sustava. Što više informacija o Web aplikaciji napadač sazna, to je veća vjerojatnost kompromitacije cijelog sustava.

2.5.1. Rasipanje informacija

Rasipanje informacija je sigurnosni propust kojima Web aplikacije otkrivaju osjetljive podatke kao što su programerski komentari i detaljne poruke o pogreškama, koje mogu pomoći napadaču u kompromitranju sustava. Osjetljivi podaci se najčešće mogu naći unutar HTML komentara, poruka o grešci, izvornom kodu ili jednostavno mogu biti javno dostupni korisnicima. Iako rasipanje informacije ne predstavlja značajnu povredu sigurnosti, ona služi napadaču kao vodilja pri konstrukciji napada. Stoga rasipanje informacija predstavlja rizik koji treba svesti na minimum.

Komentari u kodu i detaljne poruke o grešci predstavljaju rasipanje informacija kojima se napadaču daju informacije o strukturi mapa, strukturi SQL izjava i imena ključnih procesa korištenih od strane Web aplikacije. Programeri najčešće ostavljaju komentare unutar HTML dokumenta ili skriptnog jezika kako bi si olakšali prilikom *debugging-a* i integracije sustava. Ove informacije mogu varirati od jednostavnih detalja kako određena skripta radi, a u najgorem slučaju, mogu čak otkriti korisnička imena i lozinke unutar ispitne faze razvoja Web aplikacije.

Pod rasipanje informacija se također podrazumijevaju “tajni” podaci koji nisu adekvatno zaštićeni od strane Web aplikacije. Ovi podaci su različiti brojevi računa ili osobni korisnički podaci (JMBG, broj X-ice itd.). Kako bi se spriječilo rasipanje informacija, sve bitne podatke je potrebno dodatno zaštiti, pa čak sakriti od pogleda samog korisnika - vlasnika te povjerljive informacije. Broj kreditne kartice je primjer osobnog korisničkog podatka koji je potrebno dodatno zaštititi od rasipanja primjenom enkripcije i dodatne kontrole pristupa.

Jedan od primjera rasipanja informacija je sljedeći odsječak HTML koda:

```
<P>
<!--Ako slika primjer.jpg ne postoji, resetiraj Faramira -->
<a href=”www.fer.hr”><img src=”primjer.jpg” /></ a>
</P>
```

Iz ovog primjera je vidljivo rasipanje informacija kroz nepotrebni programerski komentar. Naime, programer je unutar komentara naveo ime Web poslužitelja i na taj način opskrbio napadača dodatnom informacijom koja može poslužiti za napad prema Web poslužitelju.

2.5.2. Izlistavanje mapa

Automatsko izlistavanje mapa (direktorija) je funkcija Web poslužitelja koja izlistava sve datoteke unutar tražene mape ukoliko ne postoji osnovna datoteka (`index.html`,`home.html`, `default.htm`). Izlistavanje mapa predstavlja poseban slučaj rasipanja informacija. Ako korisnik zahtjeva glavnu stranicu Web aplikacije, tada će najčešće utipkati URL poput : `http://www.primjer.ime_domene` – koristeći određeno ime domene. Web poslužitelj će procesirati njegov zahtjev i pretražiti da li u osnovnom (*root*) direktoriju postoji osnovna datoteka koju će poslati korisniku. Ukoliko ova datoteka ne postoji, poslužitelj će izlistavanjem mape korisniku poslati cjeloviti ispis direktorija. Ova funkcija je ekvivalentna s *"ls"* (Unix) ili *"dir"* (Windows) naredbe unutar određenog direktorija.

Kada Web poslužitelj otkrije sadržaj direktorija, postoji mogućnost otkrivanja datoteka unutar mapa koje nisu namijenjene javnosti. Stoga, Web administratori često koriste strategiju "Sigurnost kroz zamračivanje" (engl *Security through Obscurity)*, pri čemu pretpostavljaju da ukoliko ne postoje linkovi prema određenim datotekama, da se tim datotekama ne može pristupiti. Naravno, ta pretpostavka je netočna. Danas postoje različiti pretraživači ranjivosti koji mogu dinamički pretraživati direktorije i datoteke unutar određene Web aplikacije provodeći ispitivanje nad Web poslužiteljem.

Gledano iz perspektive sigurnosti, nepotrebno izlistavanje direktorija omogućava potencijalnom napadaču potrebne informacije za izvođenje napada na sustav. Stoga, izlistavanje mapa se koristi samo kad ne postoji druga mogućnost prikaza određenih sadržaja ili kad ne postoji rizik od napada.

2.5.3. Otkrivanje prečaca

Otkrivanje prečaca je napadačka tehnika čiji je cilj pristupiti datotekama, mapama i naredbama koje su inače izvan osnovne mape Web aplikacije. Napadač ovom tehnikom pokušava podesiti URL na takav način da će Web aplikacija izvršiti ili otkriti sadržaj određenih datoteka koje su razmještene po cijelom Web poslužitelju. To se uglavnom odnosi na sustave koje imaju HTTP sučelje prema korisniku.

Većina Web aplikacija ograničava korisnicima pristup na točno određeni dio datotečnog sustava, koji se obično naziva "web root" mapa ili osnovna mapa Web aplikacije. Osnovna mapa uglavnom sadrži datoteke koju su namijenjene korisnicima i izvršne datoteke koje su potrebne za funkcionalnost same Web aplikacije. Ali korištenjem specijalnih nizova znakova, moguć je i pristup ostalim datotekama ili naredbama bilo gdje unutar datotečnog sustava. Drugim riječima, postoji niz specijalnih znakova pomoću kojih se može stvoriti prečac.

Najosnovniji napad otkrivanja prečaca koristi "`../`" specijalni niz znakova kako bi izmjenio lokaciju određenog resursa unutar URL-a. Iako većina današnjih Web poslužitelja će otkriti ovu metodu "bijega" iz osnovnog direktorija Web aplikacije, i dalje postoje sofisticiranije metode koje napadači koriste kako bi izbjegli sigurnosne filtere. Te metode variraju od *Unicode-encoding*-a gdje se umjesto "`../`" znakova koriste "`..%u2216`" , *URL-encoding*-a ("`%2e%2e%2f`") i dvostrukog *URL-encoding*-a ("`..%255c`").

Unatoč prevenciji prečaca koju je može osigurati funkcijama Web poslužitelja, Web aplikacije i dalje mogu biti ranjive na ovu vrstu napada. Napad je uglavnom prema usmjeren Web aplikacijama koje loše rukuju ulaznim korisničkim podacima. U ovoj varijanti napada, za vrijednost izvornog URL parametra, napadač postavlja ime određene datoteke u kojoj je sadržana aplikacijska skripta. Pošto će se ta datoteka interpretirati kao tekstualna datoteka, rezultat ovog napada će biti otkrivanje izvornog koda datoteke.

Primjeri otkrivanja prečaca na Web poslužitelju su prikazani sljedećim odsječkom:

```
http://primjer/../../../../../temp/dat
http://primjer/..%255c..%255c..%255ctemp/dat
http://primjer/..%u2216..%u2216temp/dat
```

U ovom primjeru su prikazane tri vrste napada. Prvi je osnovni, koristeći "`../`" specijalni niz znakova. Drugi i treći se koriste *encoding-e,* dvostruki *URL-encoding* i *Unicode-encoding*.

Sljedeći primjer prikazuje tkrivanje prečaca na Web aplikaciji:

Izvorni link: `http://primjer/temp.cgi?home=index.htm`

Napad: `http://primjer/temp.cgi?home=temp.cgi`

U ovom primjeru, Web aplikacija će otkriti izvorni kod `temp.cgi` datoteke jer se `home` varijabla koristi za prikaz sadržaja u obliku HTML dokumenta.

2.5.4. Predviđanje lokacije resursa

Predviđanje lokacije resursa je napadačka tehnika koja se koristi kako bi se otkrili skriveni sadržaji ili funkcionalnosti Web aplikacije. Tehnika se uglavnom svodi na grubo pretraživanje sadržaja koji nije namijenjen za javno prikazivanje. Najčešće mete su privremene (*temp*), *backup*, konfiguracijske ili obične datoteke. Datoteke mogu sadržavati bitne informacije o programskoj strukturi Web aplikacije, informacije o bazi podataka, lozinke, imena računala ili čak popis ranjivosti (*buggova*). Ova metoda može biti vrlo uspješna iz razloga što skrivene datoteke često imaju posebne konvencije za nazive i nalaze se na standardnim lokacijama.

Napadači najčešće izrađuju zahtjeve za određenim datoteka i mapama, te zahtjev šalju nekom javnom Web poslužitelju. Postojanje pojedinih resursa se može utvrditi analizirajući HTTP statusne kodove dobivene kao odgovor od Web poslužitelja.

Primjer su sljedeće varijacije pretraživanja:

- Slijepo pretraživanje određenih datoteka i direktorija:

  ```
  /admin/
  /backup/
  /logs/
  /primjer.cgi
  ```

- Dodavanja ekstenzija već postojećim datotekama, npr. `test`:

  ```
  /test.asp
  /test.bak
  /test.tmp
  ```

2.6. Logičke ranjivosti

Logičke ranjivosti dozvoljavaju napadačima korištenje napadačkih tehnika koje su namijenjene narušavanju ili zloupotrebi poslovnog toka (engl. *workflow*) Web aplikacije. Pod poslovnim tokom ili aplikacijskom logikom se podrazumijeva proceduralni tok koji se koristi u Web aplikaciji kako bi se izvela određena radnja. Postoje razni primjere aplikacijske logike, a najpoznatiji su: proces za obnovu lozinke, registracija računa ili on-line kupovina. Web aplikacija zahtijeva od korisnika da točno i precizno izvodi određeni proces u koracima kako bi završio pojedinu radnju. S druge strane, napadačeva namjera je zaobići ili zlouporabiti ovaj proces kako bi naštetio Web aplikaciji i njenim korisnicima.

2.6.1. Zloupotreba funkcionalnosti

Zloupotreba funkcionalnosti je napadačka tehnika koja se koristi resursima i funkcionalnošću Web aplikacije kako bi se iskoristio, prevario ili promijenio njen kontrolni mehanizam. Neke funkcionalnosti Web aplikacije, pa čak i sigurnosne mjere je moguće narušiti i tako izazvati neočekivano ponašanje Web aplikacije. Ukoliko postoji i najmanja mogućnost narušavanja, napadač može ugroziti drugog korisnika ili prevariti cijeli sustav. Razina narušavanja ovisi od aplikacije do aplikacije.

Ova vrsta napada se najčešće isprepliće s ostalim vrstama napada. Na primjer, napadač koristi XSS napad kako bi ubacio štetnu skriptu u *Web-chat* aplikaciju i tada koristi ugrađene funkcije za propagiranje zloćudnog koda prema drugim aplikacijama. Upravo se taj način propagacije često koristi, pa možemo reći da se zloupotreba funkcionalnosti koristi i kao multipliciranje napada (engl *force multiplier*). Evo još nekoliko primjera:

- Korištenje tražilice kako bi se pristupilo zaštićenim datotekama izvan osnovne mape
- Iskorištavanje datotečnog *upload* sustava zamjenu kritičnih konfiguracijskih datoteka
- Izvođenje DoS napada nad sustavom prijave korisnika koristeći ispravna korisnička imena i loše lozinke kako bi se blokirao pristup određenim korisnicima (nakon određenom broja pokušaja Web aplikacija onemogući pristup korisniku).

Općenito, svi efektivni napadi prema aplikacijama su vezani uz ugrožavanje i zloupotrebu funkcionalnosti. Specijalno u našem slučaju, opisuju se napadi koji ruše Web aplikacije s malo ili bez ikakvi izmjena izvornih aplikacijskih funkcija.

Jedan od zanimljivijih primjera zloupotrebe funkcionalnosti je zabilježen kod *Smarwin CyberOffice* Web aplikacije koja koristi "kolica za kupnju" (engl *Shopping cart*) kako bi korisnicima omogućila kupovinu određenih artikala. Naime, napadač je na neočekivani način mogao izmijeniti ponašanje Web aplikacije mijenjajući vrijednost skrivenog polja unutar forme koju koristi Web aplikacija. Forma bi se normalno skinula, izmijenila i ponovno poslala aplikaciji s cijenama postavljenima prema napadačevim željama.

2.6.2. Uskraćivanje usluge

Uskraćivanje usluge *ili Denial of Service* (*DoS*) napadi su napadačke tehnike čija je namjera onemogućiti normalni rad Web aplikacije ili korisnički pristup Web aplikaciji. Uspješnost ovih napada ovisi o trošenju vitalnih resursa sustava, pronalasku i korištenju ranjivosti ili zlouporabi funkcionalnosti.

Čest slučaj je trošenje sistemskih resursa; kada se dosegne maksimalno korištenje određenog resursa (CPU, memorija, disk, mrežno sučelje) tada se korisniku onemogućava pristup i rad Web aplikaciji. U ovom slučaju govorimo o *DoS* napadu na mrežnom sloju koji koristi veliki broj mrežnih veza za "potapanje" sustava.

Većina današnjih Web aplikacija se temelji na okruženju koje čini Web poslužitelj, poslužitelj baze podataka i autentifikacijski poslužitelj. Posebna pažnja se posvećuje *DoS* napadima na aplikacijskom sloju koji su usmjereni prema navedenim neovisnim komponentama. Razlog tomu je, što ih je puno lakše izvesti nego one na mrežnom sloju. Neki od primjera DoS napada:

- *DoS* napad usmjeren prema korisniku. Napadač će se učestalo pokušavati prijaviti na Web aplikaciju kao određeni korisnik, namjerno koristeći krivu lozinku. Konačno, aplikacijski autentifikacijski proces će "zaključati" pristup korisniku.
- *DoS* napad prema poslužitelju baze podataka. Napadač uz pomoć *SQL* ubacivanja modificira bazu podataka na način da sustav postane nestabilan (npr obriše sve podatke).
- *DoS* napad prema Web poslužitelju. Napadač koristi napad vezan uz prelijevanje spremnika kako bi srušio procese Web poslužitelja i kako se sustavu ne bi moglo pristupiti određeno vrijeme.

2.6.3. Napadi automatiziranim procesima

Napadi automatiziranim procesima nastupaju kada Web aplikacija omogući napadaču izvođenje automatiziranog procesa koji bi se inače po svojoj prirodi trebao ručno izvesti od strane korisnika. Stoga, određene funkcije Web aplikacije bi trebale biti zaštićene od automatiziranih napada.

Ukoliko se ne vrši provjera, automatizirani programi–roboti ili napadači mogu uzastopno ispitivati funkcionalnost Web aplikacije kako bi narušili ili prevarili sustav. Automatizirani roboti mogu biti podešeni za slanje na tisuće zahtjeva u minuti izazivajući potencijalni gubitak performansi ili onemogućavanje usluge. Preventivno, treba onemogućiti robota da prijavi tisuću korisničkih računa unutar par minuta ili da pošalje tisuću poruka na forum. Ovi procesi se moraju ograničiti samo za ljudsku upotrebu.

2.6.4. Narušavanje kontrole procesa

Narušavanje kontrole procesa nastupa kada Web aplikacija dopusti napadaču zaobilaženje ili promjenu uobičajnog kontrolnog toka aplikacije. Ova vrsta napada je česta ukoliko se korisnički stanje ne provjerava unutar određenih aplikacijskih procesa.

Kada korisnik koristi određene usluge (funkcije) Web aplikacije, tada se od Web aplikacije očekuje da korisnika provede kroz određenu sekvencu događaja. Ukoliko unutar sekvence događaja, korisnik neispravno izvrši neki događaj ili izvan uobičajnog redoslijeda, tada dolazi do povrede integriteta podataka. Primjeri sekvenci događaja su već spomenuti procesi u više koraka: obnova lozinke, otvaranje korisničkog računa, *on-line* kupovina, itd. Da bi ovi procesi ispravno funkcionirali, Web aplikacija mora pratiti stanje korisnika dok korisnik vrši radnje unutar njenog toka procesa. Praćenje se najčešće vrši korištenjem sjedničkog identifikatora ili skrivenih HTML polja. Ukoliko se praćenje vrši na korisničkoj strani

unutar Web preglednika, tada je potrebno dodatno provjeravati integritet podataka. Ukoliko nema provjere, postoji mogućnost zaobilaženja kontrolnog tijeka događaja promjenom trenutnog korisničkog stanja.

Primjer: *On-line* Web aplikacija za prodaju (*shopping cart*). Svaki korisnik koji kupi proizvod A će dobiti popust, dok za proizvod B nema popusta. Korisnikova želja je da dobije popust za proizvod B. Kontrola procesa će se narušiti na sljedeći način:

1. Korisnik želi kupiti proizvod B s popustom, te u kolica ubacuje proizvode A i B u košaricu.
2. Korisnik ulazi u proces provjere gdje mu se registira popust.
3. Korisnik se vraća natrag iz procesa provjere, izbacuje proizvod A iz košarice.
4. Korisnik ponovno ulazi u proces provjere, pritom zadržavajući popust ostvaren u prijašnjem procesu provjere (s proizvodom A u košarici) i prevarom postiže nižu cijenu proizvoda B.

2.7. Zaštita i prevencija ranjivosti Web aplikacija

Sigurnost Web aplikacija u današnje vrijeme u velikoj mjeri ovisi o dizajnu Web aplikacije. Stoga, velika odgovornost je na programerima koji uz pisanje koda aplikacije moraju dodatnu pažnju obratiti i na sigurnost. Tri glavna razloga dizajniranja ranjivih Web aplikacije su:

- Mnoge Web aplikacije jednostavno nisu dizajnirane da se odupru neprijateljskom okruženju. Sigurnost u ovom slučaju nije prioritet za aplikacijskog programera; programer većinu svoje energije i vremena posvećuje sadržaju i unapređivanju pouzdanosti aplikacije.
- Mnogi programeri Web aplikacija nisu educirani za pisanje sigurnog koda. Razlog tomu je praćenje, odnosno zaostajanje programera za najnovijim operacijskih sustavima i razvojnim okruženjima. Izučavanje mrežne sigurnosti može biti naporan posao koji iziskuje znatno vrijeme potrebno za dodatne analize napada i smišljanje njihove prevencije.
- Aplikacijski programeri su ljudi, a ljudi čine pogreške. Čak i programeri kojima je sigurnost Web aplikacija prioritet, ponekad zaborave oprezno parsirati korisnički unesene podatke ili pohrane tajnu informaciju na mjesto gdje ta informacija može biti kompromitirana.

Svaka Web aplikacija može imati sigurnosni propust koji je rezultat najobičnijih programerskih grešaka, te iz dana u dan se otkrivaju nove vrste ranjivosti. U ovom poglavlju već smo opisali većinu sigurnosnih propusta i napada. Potrebno je još jednom navesti glavne predstavnike iz određenih klasa napada i način njihove prevencije:

- Iz klasa napada vezanih uz autentifikaciju, glavni predstavnik je *Brute force* napad čija je namjera, učestalim ponavljanjem doći do korisničke lozinke i tako kompromitirati proces autentifikacije. Stoga, programeri aplikacija moraju voditi računa o dodatnim zaštitama korisničkih lozinki kako bi se zaštitile od otkrivanja i nagađanja lozinke od strane napadača.

- Iz klasa napada vezanih uz autorizaciju, naglasak je napadačkim radnjama koje ugrožavaju *sjedničke identifikatore*, tj. nagađanje, kontrolu trajanja usluge i fiksaciju usluge. Ovaj problem se pokušava riješiti s dodatnim policama sigurnosti kao što je polica WWW Consortium-a (W3C), *Platform for Privacy Preferences* (*P3P*) . *P3P* koristi XML datoteku kako bi se opširno opisao način na koji Web aplikacija rukuje privatnim korisničkim podacima tijekom i poslije korištenje usluge.
- Iz klase napada na korisničku stranu, glavni predstavnik je XSS napad koji omogućava izvršavanja štetnog napadačkog koda u korisničkom Web pregledniku. Pošto Web aplikacija prosljeđuje ovaj kod prema korisniku, dodatna odgovornost je na programerima i administratorima Web aplikacije pri filtriranju zahtjeva i traženju skrivenih skripti.
- Iz klase napada vezanih uz izvršavanje naredbi, glavni predstavnici su prelijevanje memorijskog spremnika i *SQL ubacivanje*. Prelijevanjem spremnika, napadač pokušava destabilizirati Web aplikaciju ugrožavanjem memorije, dok *SQL ubacivanjem* želi preuzeti kontrolu nad bazom podataka koju koristi Web aplikacija. Obje vrste napada je moguće spriječiti dodatnom validacijom korisnički unesenih podataka od strane Web aplikacije. Programeri pri pisanju aplikacija moraju paziti na posebne znakove i naredbe koji se interpretiraju od strane Web poslužitelja, operacijskog sustava i baze podataka.
- Rasipanje vitalnih informacija mora biti sprječeno u bilo kojem obliku. Administrator, korištenjem dodatnih razina zaštite datoteka i direktorija, onemogućavanjem ispisa sadržaja direktorija, onemogućavanjem prečaca, mora stvoriti čvrstu strukturu Web aplikacije koja će spriječiti napadača da dođe do povjerljivih informacija.
- Iz klase logičkih napada, glavni predstavnik je napad vezan uz uskraćivanje usluge *ili DoS* napad . *DoS* napad "ruši" sustav Web aplikacije onemogućavajući korisniku pristup. Kako je DoS napad najčešće posljedica drugih napadačkih tehnika, prevencija te određene vrste napada sprečava i rezultirajući DoS napad.

Visoki razina sigurnosti Web aplikacije se također može postići ispravnom strategijama pisanja Web aplikacije. Jedna od najpoznatijih strategija je *SD3* strategija (engl. *secure by design, secure by default, secure in deployment*). Neke od osobine ove strategije su:

- Pisanje koda sigurnim principima.
- Jednostruka ljudska pogreška neće ugroziti sigurnost cijele aplikacije. Primjerice, prilikom pisanja koda, korisnički uneseni podaci će se parsirati i provjeravati unutar svake funkcije koja prima korisničke podatke kako bi se spriječio sigurnosni propust.
- Postojanje sustava za nadgledanje i detekciju kako bi se detektirali napadi. Na ovaj način se može otkriti identitet korisnika koji je pokušao izvesti napad i spriječiti da to čini u budućnosti. Dodatno, sustav će otkriti način na koji je napad izveden te izvijestiti programere. Programeri će analizirati napad, te izdati novu verziju Web aplikacije kako bi zaštitili druge korisnike od napada. Ovim postupkom postiže se dugoročna sigurnost Web aplikacije.

Danas najčešće korištena metoda zaštite Web aplikacija je korištenje različitih alata za identifikaciju ranjivosti. Nakon identifikacije ranjivosti, programer intervenira promjenom

osnovnog koda aplikacije kako bi ispravio sigurnosne propuste. Više o alatima za ispitivanje sigurnosti Web aplikacija u sljedećem poglavlju.

3. Alati za otkrivanje ranjivosti Web aplikacija

Organizacije koje za svoje poslovne procese koriste Web aplikacije, susreću se s sve većim problemom zaštite Web aplikacija. Čak 95% organizacija su bile žrtve barem jednog sigurnosnog incidenta vezanog uz Web aplikacije [20.]. Stoga, s svakim danom sve se više uzimaju u obzir različite metode zaštite Web aplikacija i korisničkih podataka.

Web aplikacije nije moguće zaštiti nekim uobičajnim metodama i kontrolama koje se koriste za sigurnost na nižim slojevima mreže i aplikacijskom sloju. Tu se prvenstveno misli na različite vrste vatrozida, aplikacijske posrednike (engl. proxy) ili uvođenje SSL-a. Iako postoji mnoštvo drugih obrambenih taktika, rijetko koja se je pokazala u potpunosti uspješnom. Jedan od djelotvornijih načina zaštite je pronalazak i identifikacija svih ranjivosti u Web aplikaciji, te njihova eliminacija.

Kako bi se pronašle i identificirale ranjivosti, koriste se različiti alati za ispitivanje sigurnosti. Alati se po načinu rada mogu podijeliti u dvije skupine:

- alati za ručno otkrivanje (engl. *Pen testing tools*) i
- automatizirani alati (engl. *Scanning tools*).

Odabir vrste alata ovisi o vrstama ranjivosti koje se žele otkriti i identificirati. Automatizirani alati se uglavnom koriste za otkrivanje tehničkih ranjivosti poput SQL ubacivanja i XSS napada, dok se alati za ručno otkrivanje koriste za otkrivanje logičkih ranjivosti poput zloupotrebe funkcionalnosti ili nedovoljne autentifikacije/autorizacije.

Drugi važan kriterij podjele alata je cijena, tako da imamo skupe komercijalne i besplatne alate otvorenog koda. Komercijalni alati su moćniji, ali nisu baš dostupni običnom korisniku. S druge strane, alati otvorenog koda su dostupniji, ali nisu ravni komercijalnima automatiziranim alatima.

U nastavku poglavlja, ukratko će se opisati značajke automatiziranog i ručnog otkrivanja, te će se opisati glavni predstavnici za svaku skupinu alata. Promatrati će se arhitektura, funkcionalnost, te mogućnosti analize i testiranja. Ukoliko je alat modulariziran, opisati će se glavni moduli i njihova funkcionalnost. Naglasak je na alatima otvorenog koda.

3.1. Alati za ručno otkrivanje ranjivosti

Ručno otkrivanje ranjivosti je najstarija metoda provjere sigurnosti Web aplikacija. Nekada, ali i danas u manjoj mjeri, programeri u procesu razvoja Web aplikacija, koriste alate za ručno otkrivanje kako bi ispitali Web aplikacije. S vremenom, kompleksnost Web aplikacija i frekvencija napada se povećala, pa se javila potreba za specijalistima poznatim kao *Pen-testers* koji su bili isključivo zaduženi za otkrivanje ranjivosti. Specijalisti imaju sposobnost zaključivanja na temelju manjeg skupa činjenica i samim time brže utvrđuju postojanje ranjivosti u poslovnim procesima Web aplikacije od automatiziranih alata. S druge strane,specijalist nije u mogućnosti provoditi iscrpne i temeljite testove za velik broj različitih ranjivosti, pa se to prepušta automatiziranim alatima.

Pri ručnom otkrivanju, uobičajno je testiranje korištenjem principa bijele kutije. Princip bijele kutije se temelji na pretpostavci da specijalist poznaje osnovni kod Web aplikacije i njene poslovne procese (engl. workflow), kako bi lakše pronašao ranjivosti u Web aplikaciji. Vođen ovim principom, specijalist analizira osnovni kod i poslovne procese u potrazi za određenim sigurnosnim rupama i *buggovima*. Nakon pronalaska sigurnosne rupe, specijalist

točno zna koji dio Web aplikacije je ranjiv i kako ga ugroziti, odnosno otkriva i identificira ranjivost.

Iako se danas uglavnom koriste automatizirani alati, određeni procesi i vrste ranjivosti u modernim aplikacijama se ne mogu otkriti automatiziranima alatima pa je i dalje nužno ručno otkrivanje. Tu se prvenstveno misli na logičke procese poput poslovnih procesa Web aplikacije i i različite vrste logičkih ranjivosti. Logičke ranjivosti se otkrivanju razumijevanjem rada Web aplikacije, odnosno promatranjem logičkog poslovnog toka Web aplikacije. Automatizirani alati mogu prolaziti kroz različite dijelove Web aplikacije, ali ne mogu zaključiti na koji način su ti dijelovi logički povezani. Jedino specijalisti, ukoliko razumiju logički tok Web aplikacije, su u stanju manipulirati tokom kako bi ustanovili postojanje ranjivosti. Primjerice, aplikacija upućuje korisnika od točke A kroz točku B do točke C, gdje je točka B sigurnosna validacijska točka. Promatranjem i analizom toka aplikacije, specijalist može otkriti da se iz točke A može direktno doći do točke C, odnosno zaobići sigurnosnu točku B.

Najčešće ranjivosti Web aplikacija koje se otkrivaju ručnim ispitivanjem su:

- mogućnost modifikacije cijene proizvoda,
- krađa identiteta korisnika,
- stvaranje lažnog korisničkog računa,
- manipulacija poslovnom logikom,
- povećanje korisničkih privilegija i
- ilegalni prijenos novčanih sredstava.

Glavni predstavnik alata za ručno otkrivanje je OWASP-ov WebScarab [23.].

3.1.1. WebScarab

WebScarab je OWASP-ov besplatni alat koji služi kao pomoć programerima i specijalistima za sigurnost Web aplikacija pri razumijevanju procesa i funkcionalnosti Web aplikacije, te identifikaciji mogućih ranjivosti WebScarab je alat otvorenog koda napisan u Javi, pa se može koristiti na Windows i Linux platformi. Cjelokupni osnovni kod dostupan je na OWASP-ovim Web stranicama. Usprkos činjenici da je besplatan, WebScarab je jedan od boljih i popularnijih alata za ručno testiranje Web aplikacija.

WebScarab se uglavnom koristi kao posrednik (engl. proxy) koji presreće promet između korisnika i Web aplikacije. Posrednički način rada omogućava pregled, kontrolu i promjenu korisničkih zahtjevima prije nego što su poslani prema poslužitelju Web aplikacije. Također se mogu pregledati odgovori poslani od strane Web poslužitelja, prije nego što stignu do korisnikovog Web preglednika. Posrednik može nagledati HTTP i HTTPS promet, pri čemu se svaka razmjena paketa pohranjuje u komunikacijsku listu kako bi se dodatno mogla pregledati i analizirati.

WebScarab se temelji na modularnoj arhitekturi. Postoje dvije osnovne vrste modula:

1. moduli za ostvarivanje komunikacije i
2. moduli za analizu komunikacije.

Moduli za ostvarivanje komunikacije koriste specifične metode kako bi donijeli odluku koje resurse zahtijevati od Web poslužitelja i kako parametrizirati zaglavlja HTTP ili HTTPs zahtjeva, te šalju zahtjev prema Web poslužitelju. Nakon primitka odgovora od Web poslužitelja, moduli mogu provesti izmjenu odgovora i odlučiti da li će se odgovor pohraniti u komunikacijsku listu. Kada se par zahtjev-odgovor pohrani u komunikacijsku listu, tada se daje na raspolaganje modulima za analizu komunikacije.

Trenutna verzija *WebScarab*-a sadrži sljedeće module:

- posrednik (engl. *Proxy*),
- modul za ručne zahtjeve,
- modul za snimanje lokacije resursa (engl. *Spider),*
- analizator sjedničkih identifikatora,
- skriptni modul,
- fragmentni modul,
- modul za usporedbu i
- modul za ubacivanje (engl. *Fuzzer).*

Posredničkim modulom se promatra promet između korisničkog Web preglednika i Web poslužitelja. Kao što je već rečeno, posrednički modul može promatrati HTTP i kriptirani HTTPS promet korištenjem SSL veze između *WebScarab*-a i Web preglednika. Posrednički modul također omogućava potencijalnom operatoru intervenciju u zahtjeve i odgovore koji prolaze kroz WebScarab.

Ručno presretanje se koristi kako bi se presreli, dodatno istražili, i promijenili prije ponovnog slanja zahtjevi Web preglednika i odgovori od Web poslužitelja. Primjerice, ručno presretanje je posebno korisno kada korisnik želi ispuniti formu i poslati je Web poslužitelju, ali JavaScript validacija odbija odbija formu zbog krivo unesenih vrijednosti. Ručnim presretanjem zahtjev se može izmijeniti nakon procesa validacije, odnosno može se poslati forma s krivo unesenim vrijednostima.

Modul za ručne zahtjeve omogućava korisniku ručno stvaranje zahtjeva prema Web poslužitelju, te korištenje starih zahtjeva iz komunikacijske liste za njihovo ponovno slanje ili moguće izmjene. Ovaj modul može i prikupljati identifikatore sjednice kako bi se koristili u budućim zahtjevima.

Modul za snimanje lokacije resursa analizira odgovore dobivene od strane Web poslužitelja kako bi identificirao linkove iz tijela poruke ili iz lokacijskog zaglavlja odgovora. Ako se pronađeni URL nije koristio u prethodnim komunikacijama, on se dodaje u stablo kojim se prate sve veze između svih prikupljenih linkova.Dodatno, svaki URL može razviti svoje podstablo pretraživanjem linkova iz njemu pripadnih dokumenata. Drugim riječima, modul za snimanje lokacija resursa organizira sve linkove Web aplikacije u stablastu strukturu kako bi se opisale veze između pojedinih dokumenata.

Analizator sjedničkih identifikatora skuplja i analizira određeni broj sjedničkih identifikatora ili URL parametara kako bi se vizualno odredio stupanj slučajnosti i predvidljivosti. Ovaj modul označava sve sjedničke identifikatore s vremenskom oznakom

njihovog prikupljanja te izračunava numeričke vrijednosti sjedničkih identifikatora. Na temelju vrijednosti crta se vremenski graf promjene vrijednosti pomoću kojeg se može identificirati uzorak generiranja sjedničkih identifikatora.

Skriptni modul omogućava korisnicima stvaranje skripti za ispitivanje korištenjem BSF (Bean Scripting Framework) jezika. BSF skripte koriste *WebScarab*-ove Java objekte kako bi stvorili posebne zahtjeve i poslali ih prema Web poslužitelju, te proveli analizu odgovora.

Fragmentni modul parsira HTML kako bi izdvojio skripte i komentare. Ovaj modul se pretežno koristi za traženje skrivenih polja i pregledavanje HTML koda.

Modul za usporedbu omogućava korisniku procjenu razlike između određenog broja odgovora. Procjena se korisna kada korisnik pošalje određeni broj zahtjeva na neki URL, po mogućnosti koristeći skriptni modul, te želi provesti evaluaciju rezultata. Određeni broj odgovora se može eliminirati, što znatno ubrzava analizu.

Modul za ubacivanje omogućava korištenje različitih kombinacija parametara. Pomoću ovoga modula može se promijeniti metoda slanja zahtjeva, osnovni URL, verzija zahtjeva i lista parametara kako bi se razotkrile neispravne provjere parametara koje mogu dovesti do ranjivosti poput ubacivanja napadačkog koda i SQL ubacivanja.

Osnovni postupak otkrivanja ranjivosti korištenjem WebScarab-a se može podijeliti u dvije faze. U prvoj fazi, provodi se analiza. U fazi analize preporuča se korištenje posrednika, fragmentnog modula, skriptnog modula i modula za otkrivanje lokacije resursa. Posrednik je vrlo koristan u analizi komunikacije, pogotovo pri istraživanju strukture zahtjeva i odgovora. Fragmentnim modulom se mogu otkriti ranjivosti vezane uz otkrivanje informacija uzrokovane skriptama i komentarima. Modul za otkrivanje lokacije će pretražiti cijelo stablo linkova Web aplikacija, i čak pronaći skrivene datoteka. Cilj analize je skupiti što više informacija koje se mogu iskoristiti u fazi testiranja.

Druga faza je faza testiranja. U ovoj fazi su posebno korisni sljedeći moduli: modul za ručne zahtjeve, analizator sjedničkih identifikatora i modul za ubacivanje. Modul za ručne zahjeve se koristi za stvaranje specifičnih zatjeva i za manipulaciju sjedničkih identifikatora. Također se može koristiti za otkrivanje ranjivosti uzrokovane HTTP razdvajanjem odgovora. Analizator sjedničkih identifikatora skuplja i vrši pregled dovoljne količine sjedničkih identifikatora kako bi otkrio ranjivosti vezane uz predviđanje identifikatora ili mogućnost napada grubom silom. Modul za ubacivanje je koristan za otkrivanje ranjivosti vezanih uz SQL i LDAP ubacivanje, XSS ranjivosti i nedovoljne provjere valjanosti procesa.

3.1.2. Primjer ručnog otkrivanja ranjivosti Web aplikacija

U sljedećem primjeru će se pokazati otkrivanje ranjivosti u Mambo 4.5.3 Web aplikaciji korištenjem *WebScarab* alata za ručno ispitivanje.

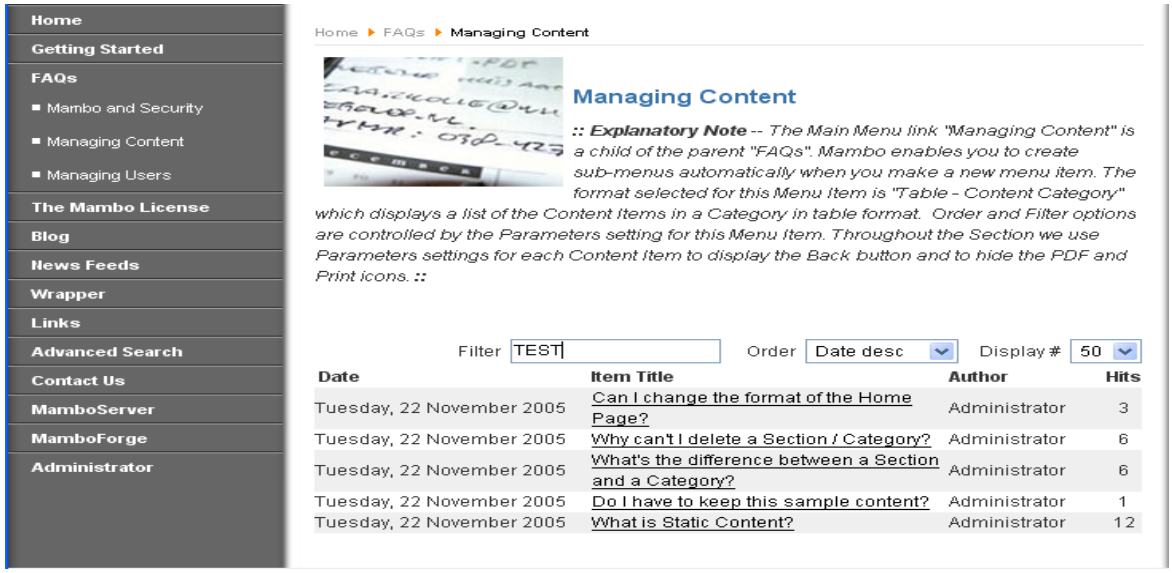
Kao što je već spomenuto, u postupku ručnog otkrivanje ranjivosti koristi se princip bijele kutije. Slijedeći taj princip, provodi se analiza osjetljivih dijelova osnovnog koda Mambo 4.5.3 CMS-a. Osjetljivi dijelovi koda su one procedure u kojima se rukuje parametrima unešenim od strane korisnika. Analizom koda otkriven je sljedeći ranjivi programski odsječak prikazan slikom 3.1 .

```
$filter = trim( mosGetParam( $_POST, 'filter', '' ) );
$filter = strtolower( $filter );
$and = '';
if ( $filter ) {
   if ( $params->get( 'filter' ) ) {
       switch ( $params->get( 'filter_type' ) ) {
         case 'title':
             $and = "\n AND LOWER( a.title ) LIKE '%". $filter "%'";break;
         case 'author':
             $and = "\n AND ( ( LOWER( u.name ) LIKE '%". $filter."%' ) OR
              (LOWER( a.created_by_alias ) LIKE '%". $filter ."%'";break;
        case 'hits':
               $and = "\n AND a.hits LIKE '%". $filter ."%'";break;

            }
      }
}
```

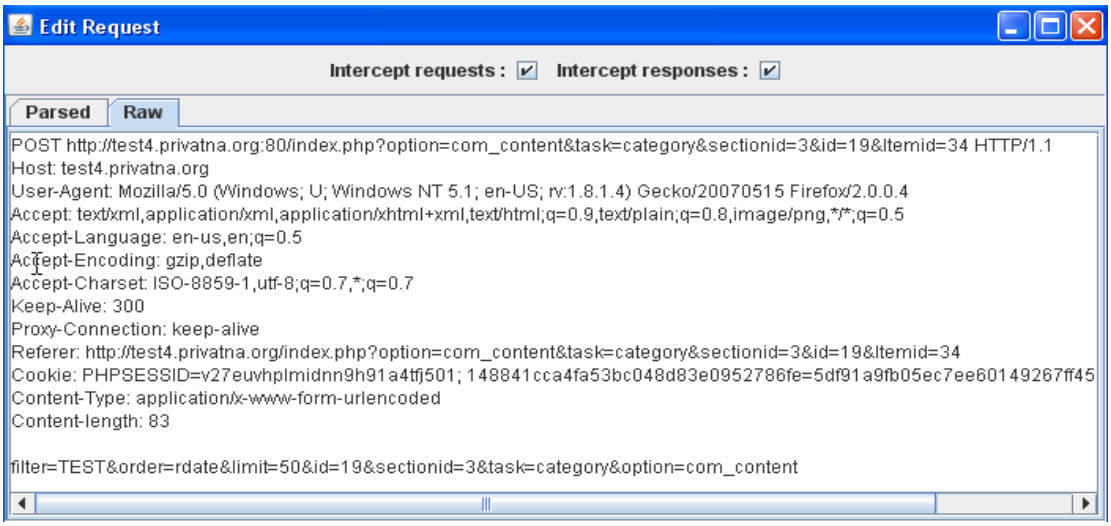
Slika 3.1. Ranjivi programski odsječak

Iz odsječka se vidi da varijabla `filter` nije adekvatno provjerena prije uvrštavanja u SQL izjavu. Stoga se sumnja na postojanje ranjivosti vezane uz SQL ubacivanje.Varijabla `filter` se koristi za filtiranje članaka u sekciji FAQ. Na slici 3.2 je prikazano upisivanje riječi TEST koja se koristi kao filter.



Slika 3.2. Unošenje filter parametra

Nakon unošenje filtera, korištenjem WebScarab-a presreće se HTTP zahtjev s pripadnim sadržajem (slika 3.3).

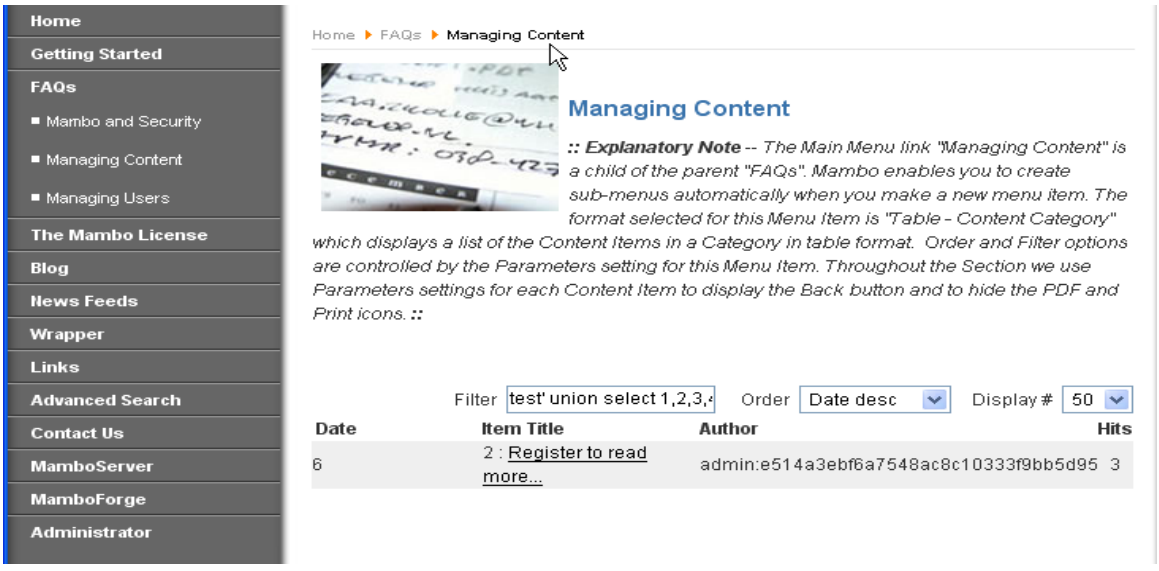


Slika 3.3. Presretanje HTTP zahtjeva korištenjem WebScaraba

Zadnja linija zahtjeva se preuredi na način da joj se doda nastavak kojim se realizira SQL ubacivanje:

```
filter=TEST' UNION SELECT  1,2,CONCAT(username,CHAR(58),password),6,7,
      FROM mos_users
      WHERE 1/&order=rdate &limit=50& id=19 &sectionid=3&task=category
              &option=com_content
```

Nastavak mijenja strukturu osnovne izjave kako bi se ispisali svi autori i njihovi sažeci lozinki. Specijalno skrojeni zahtjev se zatim proslijedi prema Web aplikaciji. Odgovor od Web aplikacije je prikazan na slici 3.4.



Slika 3.4. Odgovor s otkrivenim informacijama

Vidljivo je da se za razliku od gornje slike, u stupcu Author nalazi par autorov ID : sažetak autorove lozinke. Prema tome, SQL ubacivanje je uspješno izvedeno i pronađena je ranjivost Web aplikacije.

3.2. Automatizirani alati za otkrivanje ranjivosti

Prvi alati za automatizirano otkrivanje ranjivosti Web aplikacija pojavili su se krajem 90-tih godina prošlog stoljeća[22.]. Vremenski se pojava automatiziranih alata podudara s razvojem Web-a, točnije pojavom složenijih i dinamičnijih Web aplikacija. Ako se uzme u obzir kompleksnost i veličina današnjih Web aplikacija, ručno otkrivanje ranjivosti Web aplikacija bi bio iznimno zahtjevan i dugotrajan proces. Stoga, sve se više koriste automatizirani alati koji omogućavaju otkrivanje ranjivosti za relativno kratko vrijeme. Glavni cilj automatiziranih alata je automatizacija procesa otkrivanja unutarnjih struktura Web aplikacije i snimanje mogućih točaka napada kako bi se otkrile ranjivosti.

Automatizirani alati su složeni sustavi koje provode istraživanje struktura i stabla linkova, analizu i testiranje Web aplikacija u potrazi za tehničkim ranjivostima. U tehničke ranjivosti spadaju:

- XSS ranjivosti,
- ranjivosti vezane uz SQL ubacivanje,
- pronalazak prečaca,
- mogućnost izvršavanja OS naredbi,
- prelijevanje spremnika,
- izlistavanje mapa,
- otkrivanje skrivenih i *backup* datoteka i mapa i
- otkrivanje konfiguracijskih datoteka.

Automatizirani alat treba pružiti opširnost i potpunost prilikom procesa otkrivanja. Drugim riječima, dobar automatizirani alat treba imati mogućnost prikupljana velikog broja različitih informacija i treba imati veliku bazu različitih vrsta ranjivosti kako bi se adekvatno i temeljito ispitala sigurnost Web aplikacija. Proces automatiziranog ispitivanja ranjivosti se može podijeliti u sljedeće faze:

1. Pretraživanje lokacije resursa i dohvat sadržaja unutar Web aplikacije (engl. *Web spidering*) - ocrtava se interna struktura (stablo) linkova Web aplikacije i dohvaća se dostupni sadržaj (uglavnom datoteke). Utvrđuju se lokacije svih resursa i imena datoteka koji se mogu koristiti u otkrivanju ranjivosti kao što su administratorske mape, skrivene datoteke i pomoćne skripte. Provodi se detaljna analiza HTML-a, kako bi se izvukle korisne informacije poput klijentskih skripti, adresa elektroničke pošte, komentara ili skrivenih URL-a.
2. Pretraživanje grubom silom – pokušava se doći do povjerljivih podataka kako bi se dobio pristup određenom dijelu ili funkcionalnosti Web aplikacije.Cilj grube pretrage je dohvaćanje korisničke lozinke ili stjecanje sjedničkog identifikatora.
3. Ubacivanje (engl. *Fuzzing*) – proces u kojem analizira svaka forma ili parametar koji koristi Web aplikacija kako bi se otkrilo nepropisno rukovanje određenim sadržajem koje vodi do različitih vrsta ranjivosti. Cilj procesa je snimiti potencijalne sigurnosne rupe.

4. Otkrivanje ranjivosti (engl. *Vulnerability Scanning*) – kompleksna metoda koja koristi različite metode za otkrivanje pojedinih ranjivosti. Općenito, u ovom procesu se šalju "štetni" podaci kako bi se izazvala reakcija Web aplikacije koja ukazuje na ranjivost, te se vrši pravilna klasifikacija ranjivosti.

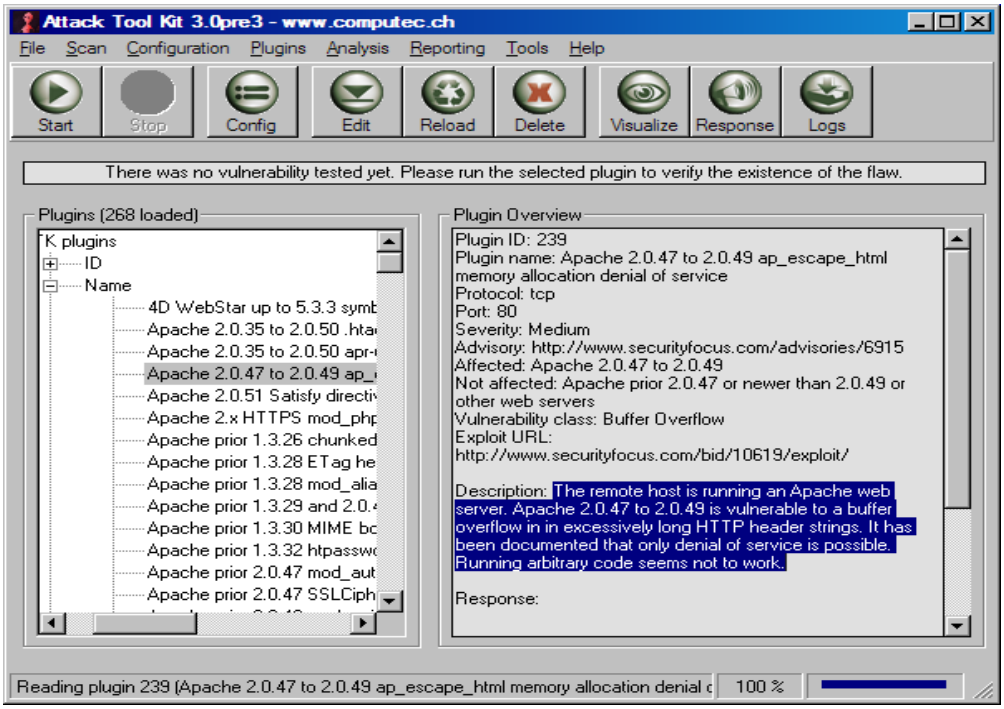
Današnji automatizirani komercijalni alati koriste pregršt metoda za otkrivanje ranjivosti te sadrže, veliku bazu ranjivosti koja se redovito ažurira. Komercijani alati su moćno rješenje koje koriste svi specijalisti pri ispitivanju sigurnosti Web aplikacija kako bi otkrili i otklonili većinu sigurnosnih propusta. Automatizirani alati otvorenog koda zaostaju za komercijalnim alatima po mogućnostima te nemaju dovoljno razvijene procese otkrivanja i opširnu bazu ranjivosti. Ipak, zbog dostupnosti široj računarskoj zajednici, u nastavku će se opisati par automatiziranih alata otvorenog koda.

3.2.1. ATK

Attack Tool Kit (ATK) [25.] je besplatni alat razvijen u Visual Basic-u za Windows platformu koji se koristi za brzo otkrivanje poznatih ranjivosti.

U kontekstu ATK-a modul je jedan test dizajniran kako bi se otkrila specifična poznata ranjivost. Trenutna verzija alata posjeduje više oko 300 modula, koji se redovno ažuriraju i daju na raspolaganje korisnicima ili se čak mogu napisati vlastiti moduli za testiranje. Moduli se pišu jednostavnim skriptnim jezikom čiji se detaljni opis može naći u dokumentaciji alata. Prednost ATK alata je upravo mogućnost jednostavne modifikacije modula, koji se čak mogu modificirati i za vrijeme izvođenja pojedinog testa.

Moduli podržavaju niz parametara uz pomoć kojih se upravlja testiranje. Svaki modul ima svoje ime, identifikator, pristupni broj (engl. *port number*), klasu ranjivosti i sigurnosni rizik ranjivosti (engl. *severity*). Traženjem po određenom parametara znatno se olakšava postupak testiranja. Na slici 3.5 je prikazan prozor putem kojeg se odabire modul za testiranje ranjivosti *Apache* poslužitelja korištenjem pretrage po imenu ranjivosti. Slika također prikazuje detaljan opis ranjivosti.



Slika 3.5. Prozor za odabir modula u ATK-u

Velika prednost ATK-a je jednostavnost korištenja. Korisnik treba samo odabrati ranjivost koju želi testirati i čekati odgovor od alata. ATK utvrđuje postojanje ranjivosti, te vizualno dočarava korisniku uspješnost napada. Prema tome, ATK je vrlo pristupačan programerima i običnim korisnicima Web aplikacija jer ne treba neko posebno znanje o sigurnosti i poslovnim procesima Web aplikacije.

ATK je odličan alat za utvrđivanje postojanja specifične poznate ranjivosti i eksploataciju pojedine ranjivosti. To ga čini korisnim za otkrivanje ranjivosti u komercijalnih Web aplikacijama, Web aplikacijama otvorenog koda i Web poslužiteljima. Također može poslužiti pri izvođenju ručnog otkrivanja. ATK nije dizajniran za automatsko testiranje Web aplikacija izrađenim prema zahtjevima korisnika i za pronalazak novih sigurnosnih rupa. Prilikom svog rada, oslanja se uglavnom na bazu znanja poznatog Web poslužitelja i Web aplikacije kojima želi pronaći ranjivosti. Za testiranje aplikacija izrađenih prema korisničkim zahtjevima, treba napisati nove module s prilagođenim opisom testa.

3.2.2. Wapiti

Wapiti [26.] je alat otvorenog koda za automatizirano otkrivanje nepoznatih vrsta ranjivosti Web aplikacija. Napisan je u Python programskom jeziku pod GPL licencom pa se može slobodno koristiti i prilagoditi korisničkim potrebama. Dijelovi Wapiti-a su korišteni u programskoj implementaciji sustava za otkrivanje vrste Web aplikacija.

Wapiti se temelji na otkrivanju ranjivosti principom crne kutije. Stoga je vrlo koristan za otkrivanje sljedećih tehničkih ranjivosti: SQL i LDAP ubacivanja, XSS, otkrivanje prečaca, loše rukovanje datotekama i izvršavanje OS naredbi.

Wapiti sadrži više modula, a najznačajniji su *webspider* i *fuzzer*. *Webspider* prikuplja informacije o Web aplikaciji prolaskom kroz sve ponuđene linkove. Rekurzivnim pretraživanjem i analizom HTML-a svakog ponuđenog URL-a, prikuplja forme i zapisuje valjane linkove. *Fuzzer* koristi informacije dobivene od *Webspider*-a kako bi locirao potencijalne ranjive točke. Potencijalne ranjive točke su parametri izvučeni iz URL-a i polja za unos unutar formi. U svaku potencijalu ranjivu točku ubacuje se predefinirani nizovi znakova kako bi utvrdila određena vrsta ranjivosti. Primjerice, koristeći znakovni niz `<script> ... </script>` se otkrivaju XSS ranjivosti a `../` otkrivaju se prečaci.

Wapiti se pokreće upisivanjem početnog URL-a ili domene Web aplikacije putem komandne linije (slika 3.6). Korisnik također može opcionalno koristi sjedničke identifikatore i posrednike u procesu automatiziranog otkrivanja ranjivosti.

```
bash-3.0$ python wapiti.py http://127.0.0.1/vuln/ -c cookies.txt -x
http://127.0.0.1/vuln/index.php?page=logout

Prikupljanje linkova i formi:.........................

Attacking urls (GET)...
-----------------------
Warning fread (article) in http://127.0.0.1/vuln/
Evil url:
http://127.0.0.1/vuln/?article=http%3A%2F%2Fwww.google.fr%2F&page=artic
les
Unix include/fread (article) in http://127.0.0.1/vuln/
Evil url:
http://127.0.0.1/vuln/?article=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F
..%2F..%2Fetc%2Fpasswd&page=articles
```

Slika 3.6. Pokretanje i rad Wapiti-a

Rad *Wapiti*-a započinje prikupljanjem informacija, odnosno prikupljanjem linkova i formi. Zatim se iz interne baze izvlače svi znakovni nizovi ranjivosti koji se ubacuju u pojedine parametre iz URL-a i polja u formi. Slika 3.7 prikazuje otkrivanje ranjivosti.

```
Attacking forms (POST)...
-------------------------
SQL Injection found with http://127.0.0.1/vuln/login.php
  and params = login=%27%22%28&password=on
  coming from http://127.0.0.1/vuln/?page=login
```

Slika 3.7. Otkrivanje ranjivosti

Ukoliko je napad uspješan, ispisat će se katogorija napada te naznačiti kritičnu formu ili kritični parametar URL-a koji predstavlja sigurnosnu rupu Web aplikacije.

4. Sustav za ispitivanje sigurnosti Web aplikacija

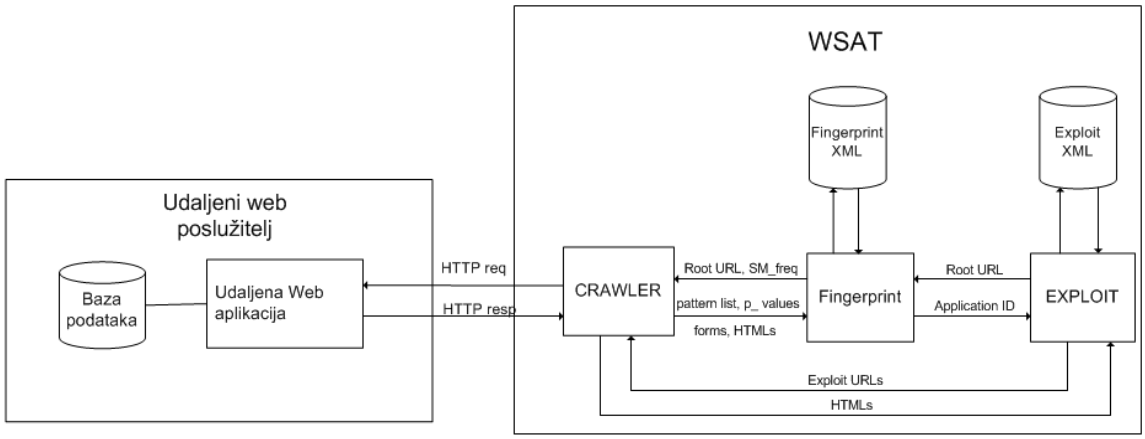
Sustav za ispitivanje sigurnosti Web aplikacija je modularni sustav pomoću kojeg se vrši proces otkrivanja ranjivosti u Web aplikaciji. U sklopu ovog diplomskog rada razmatrati će se sustav pomoću kojeg se automatizirano otkrivaju poznati sigurnosni propusti za točno određenu Web aplikaciju.

Većina komercijalnih Web aplikacija i Web aplikacija otvorenog koda,posjeduju sigurnosne rupe u svom osnovnom kodu. Sigurnosne rupe su najčešće ulazne ranjive točke kroz koje se ubacuju kompromitirajući znakovni nizovi. Tu se prvenstveno misli na parametre koji se predaju aplikaciji putem URL-a ili putem POST zahtjeva, polja forme i skrivena polja. Detaljnom analizom osnovnog koda Web aplikacija mogu se otkriti potencijalne sigurnosne rupe te se objaviti na Internetu kao ranjivosti tipične za određenu vrstu i verziju Web aplikacija. Te poznate ranjivosti su dostupne na Internetu u obliku CVE baze, na Web stranicama organizacija koje se bavi računarskom sigurnošću, ili na Web stranicama specijalista i provaljivača. Specijalisti i provaljivači najčešće otkrivaju ranjivosti automatiziranim alatima i analizom programskog koda Web aplikacije otvorenog koda. Nakon otkrivanja ranjivosti, specijalisti opisuju i klasificiraju ranjivost, te naznačuju ranjivi dio Web aplikacije (najčešće parametar) i čak opisuju cjeloviti napad pomoću kojeg se može kompromitirati Web aplikacija. Ukoliko se taj napad može formalno opisati u obliku točno određenog slijeda koraka, tada se može automatizirano utvrditi postojanje poznate vrste ranjivosti.

Programski okvir za automatizirano ispitivanje Web aplikacija se naziva WSAT – *Web Security Assessment Tool*. Kako je ispitivanje ranjivosti složen postupak, ideja je realizirati WSAT kao modularni sustav. Korištenje modularnog sustava omogućava lakšu prilagodbu postupku ispitivanja ranjivosti Web aplikacija. U pravilu, ispitivanje sigurnosti je postupak koji se sastoji od više faza, pri čemu se svaka faza može programski ostvariti modulom. Gledano iz perspektive WSAT-a, postupak ispitivanja ranjivosti Web aplikacije prolazi kroz sljedeće faze:

- prikupljanja informacija,
- određivanje vrste i Web aplikacije i
- ispitivanje poznatih ranjivosti i analiza uspješnosti napada.

Arhitektura WSAT-a, prilagođena fazama ispitivanje sigurnosti je prikazana na slici 4.1.



Slika 4.1. Arhitektura sustava za ispitivanje sigurnosti Web aplikacija – WSAT

WSAT se sastoji od sljedećih modula:

- modul za prikupljanje informacija ili *Crawler* modul,
- modul za određivanje vrste i verzije Web aplikacije ili *Fingerprint* modul i
- modul za otkrivanje ranjivosti ili *Exploit* modul.

Modul za prikupljanje informacija je zadužen komunikaciju i prikupljanje informacija iz Web aplikacije. Prikupljanje informacija se provodi analizom HTML dokumenata koji su sastavni dio Web aplikacije. *Crawler* je jedini modul koji komunicira s Web aplikacijom, tako da posjeduje sve potrebne komunikacijske programske strukture.

Modul za određivanje vrste Web aplikacije, na temelju informacija dobivenih od modula za prikupljanje informacija, utvrđuje vrstu i verziju Web aplikacije. Odluka o vrsti i verziji se donosi na temelju usporedbe preuzetih informacija s onima koji se nalaze u XML bazi podataka. Svaki XML dokument iz baze predstavlja jedan uzorak koji opisuje određenu vrstu i verziju Web aplikacije. Po točno određenim kriterijima uzorci se vrednuju, a najbolje vrednovani uzorak upućuje o kojoj se Web aplikaciji radi.

Modul za otkrivanje ranjivosti, nakon otkrivanja vrste i verzije Web aplikacija, pregledava svoju XML bazu ranjivosti kako bi pronašao već poznatu ranjivost za prethodno otkrivenu vrstu i verziju Web aplikacije. Nadalje, modul izvršava napad definiran u XML bazi ranjivosti, te vrši analizu uspješnosti napada.

U sljedećim poglavljima detaljno će se opisati gore navedeni moduli. Naglasak ovog rada je na modulu za prikupljanje informacija i na modulu za udaljeno otkrivanje vrste i verzije Web aplikacije. Modul za otkrivanje ranjivosti će se samo načelno opisati za specifični primjer ranjivosti

4.1. Modul za prikupljanje informacija

4.1.1. Komunikacija s Web aplikacijom

Jedna od glavnih zadaća modula za prikupljanje informacija je komunikacija s Web aplikacijom. Komunikacija se ostvaruje korištenjem različite komunikacijske rukovatelji (engl. *Handlers*) koji posjeduju potrebnu programsku podršku prilagođenu određenoj vrsti komunikacije. Specijalno kod WSAT-a, modul za prikupljanje informacija koristi sljedeće rukovatelje:

- HTTP i HTTPS rukovatelj,
- *Proxy* rukovatelj i
- rukovatelj sjedničkim identifikatorima.

HTTP i HTTPS rukovatelji omogućavaju komunikaciju modula za prikupljanje informacija s Web aplikacijom korištenjem HTTP-a ili sigurnijeg HTTPS-a. Ovo su temeljni načini komunikacije, prisutni u bilo kojoj vrsti komunikacije modula s Web aplikacijom. HTTP rukovatelj može koristiti dvije metode za slanje HTTP zahtjeva prema Web aplikaciji: GET i POST metodu. Razlika između ove dvije metode je u prenošenju parametera. Kod GET metode, nazivi parametara i vrijednosti parametara su sadržani unutar traženog URL-a HTTP zahtjeva, dok POST metoda zahtjeva posebno umetanje naziva parametara i njihovih vrijednosti unutar tijela HTTP zahtjeva.

Proxy rukovatelj omogućava modulu za prikupljanje informacija komunikaciju s Web aplikacijom korištenjem posrednika (engl. *proxy*). Ovaj način komunikacije se koristi prilikom testiranja ili dodatnog nadzora HTTP prometa. Na primjer, možemo koristiti WebScarab alat kao posrednik pri nadgledanju ili izmijeni sadržaja određenog HTTP paketa.

Rukovatelj sjedničkih identifikatora omogućava korištenje sjedničkih identifikatora pri komunikaciji modula za prikupljanje informacija s Web aplikacijom. Kako je HTTP protokol bez stanja, dakle ne pamti se veza između prijašnjeg i trenutnog HTTP zahtjeva, potrebno je uvesti korištenje sjedničkih identifikatora za identifikaciju korisničke sjednice. Uz rukovoditelja za korištenje sjedničkih identifikatora, modul za prikupljanje informacija također posjeduje programske metode za prikupljanje identifikatora i provjeravanje valjanosti identifikatora dobivenih od određene Web aplikacije.

Sjednički identifikator se prikuplja ispunjavanjem forme za prijavu korisnika koju je potrebno ispuniti od korisnika WSAT-a. Prikupljeni identifikator se pohranjuje u određenu datoteku, te se može naknadno koristiti. Prije svakog korištenja identifikatora, provjerava se da li je identifikator valjan, odnosno da li njegovo korištenje donosi određena prava. Provjera se vrši usporedbom HTML dokumenata dobivenih bez i s korištenjem identifikatora. Ukoliko HTML dokument dobiven HTTP zahtjevom s identifikatorom sadrži manje formi ili sadrži dodatne specifične riječi koje definiraju autoriziranog korisnika, tada je identifikator valjan. Primjerice, nakon uspješno izvršenog postupka autentifikacije/autorizacije dolazi do eliminacije forme za prijavu korisnika, te pojavljivanje specifične riječi *odjavi se!* (engl. *Logout*).

4.1.2. Prikupljanje informacija iz Web aplikacije

Koristeći postojeću komunikacijsku infrastrukturu, modul za prikupljanje informacija dohvaća HTML dokumente određene Web aplikacije kako bi izdvojio korisne informacije. Prikupljeni HTML dokumenti moraju biti dobro oblikovani i ispravni (engl. *well formed and valid*). Drugim riječima, HTML dokumenti moraju imati pravilan redoslijed ugniježđenih elemenata, svi elementi moraju biti zatvoreni (<element/>) ili (<element></element>) , zahtjeva se pravilni zapis elemenata (<ele> nije isto što i <ELE>) i zahtjeva se korištenje navodnika pri navođenju vrijednosti elemenata (<ele> "vrijednost" </ele>).

Većina HTML dokumenata dostupnih na Internetu ne zadovoljava ove uvjete, odnosno postoje greške u strukturi HTML dokumenta. Stoga je pri analizi HTML dokumenata potrebno koristiti programsku podršku koja popravlja strukturu HTML dokumenata. Sljedeći odsječak prikazuje nepravilnu strukture dijela HTML dokumenta:

```
<i><h1>heading</h1></i>
<p>new paragraph <b>bold text
<p>some more bold text
```

Korištenjem programa za uređivanje HTML dokumenata, navedena nepravilna struktura se popravlja:

```
<h1><i>heading</i></h1>
<p>new paragraph <b>bold text</b>
<p><b>some more bold text</b>
```

Vidljivo je da se pravilno postavlja pozicija elemenata <i> i <h1> te se pravilno zatvara element za podebljavanje teksta <b>.Element se mijenja, odnosno ne zatvara se s

Sljedeći odsječak također prikazuje nepravilnu strukturu HTML dokumenta:

```
<head>
 <meta http-equiv="refresh" content="0;url=http://os2.zemris.fer.hr">
</head>
```

Meta element nije pravilno zatvoren, pa će ga uređivač HTML-a promijeniti u:

```
<head>
<meta http-equiv="refresh" content="0;url=http://os2.zemris.fer.hr"/>
</head>
```

Nakon što se pravilno uredi dohvaćen HTML dokument, modul za prikupljanje informacija prolazi kroz svaki HTML element kako bi izdvojio korisne informacije. Pod korisnim informacijama prvenstveno se misli na linkove i forme, pa se posebna pozornost posvećuje <a > i <form> elementima.

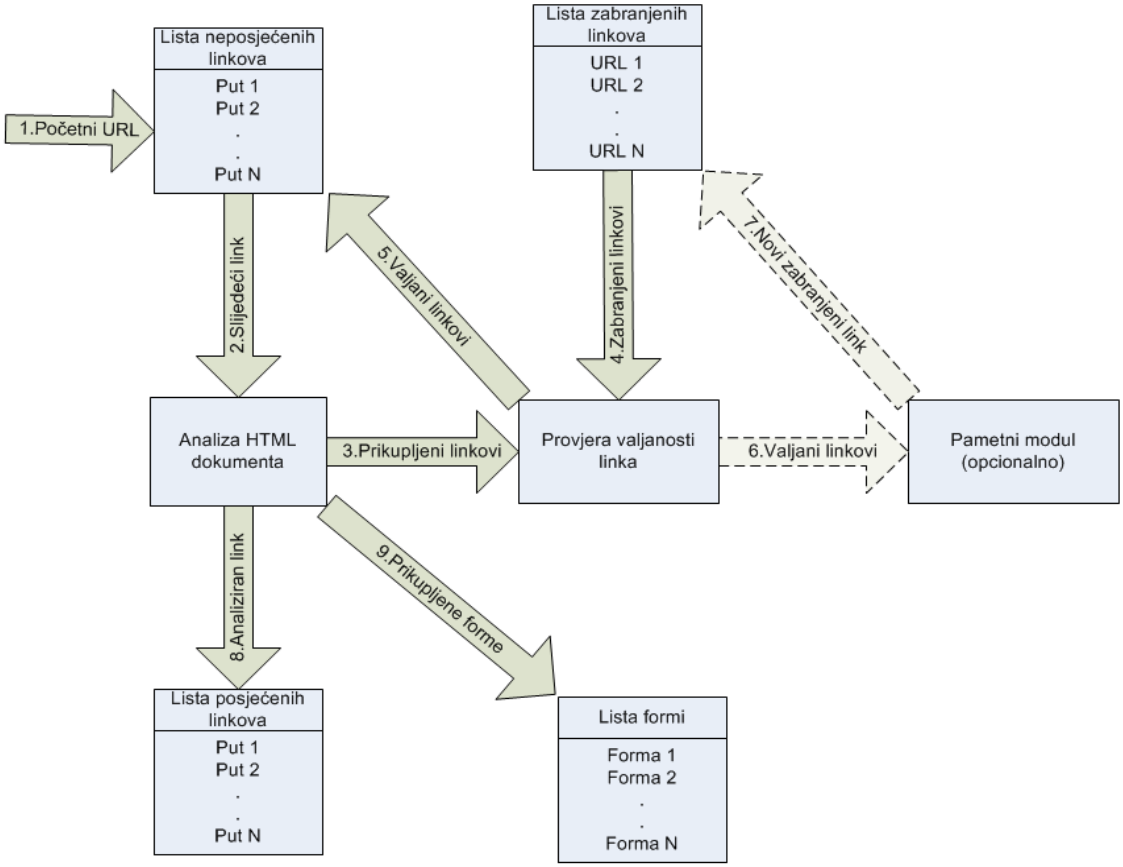
Postupak pretraživanja linkova započinje zadavanjem početnog URL-a ili domene Web aplikacije. Pri pretraživanju modul za prikupljanje informacija koristi tri liste za snimanje strukture (stabla) linkova: posjećeni linkovi, neposjećeni linkovi i zabranjeni linkovi. Na početku rada u listi neposjećenih linkova se nalazi početni URL Web aplikacije (root URL), dok su ostale liste prazne. Modul dohvaća HTML dokument s početnog URL-a, uređuje HTML dokument, te provodi analizu <a> i <meta> elemenata čije vrijednosti sadrže nove linkove. Kada se pronađe novi link, vrši se pretvorba i provjera valjanosti linka na sljedeći način:

1. Relativni URL linka se pretvara u apsolutni URL.

2. Za svaki URL se provjerava valjanost protokola. Valjani protokoli su HTTP i HTTPS.
3. Provjerava se domena URL-a. Uzimaju se u obzir URL-ovi s početno definiranom domenom.
4. Provjerava se da li se URL-om linka dohvaća dozvoljena datoteka (npr. html, php, asp, htm itd.)
5. Provjerava se da li je URL već posjećen ili da li je zabranjen

Ukoliko novi link zadovoljava gore navedene uvjete, tada se on dodaje u listu neposjećenih linkova skupa s svim svojim predhodnicima kako bi se pamtio cijeli put do linka. Put se sastoji od niza URL-a čijim se slijedom došlo do određenog konačnog linka.

Nakon što završi s analizom HTML-a s određenog URL-a, taj URL se dodaje u listu posjećenih linkova, te se postupak iterativno ponavlja za slijedeći URL u listi neposjećenih linkova. Lista posjećenih linkova sadrži liste kojima se opisuje cjelokupni put do upisanog URL-a. Lista zabranjenih linkova zadrži popis URL-ova koji se ne smiju pretraživati. Osim apsolutnog URL-a, u listu zabranjenih linkova se mogu dodati i regularni izrazi (primjerice *index.php?id=**). *Crawler* modul završava s radom kad se isprazni lista neposjećenih linkova. Cjelokupni proces prikupljanja informacija je prikazan na slici 4.2.



Slika 4.2. Rad modula za prikupljanje informacija

Prilikom pretraživanja linkova, modul za prikupljanje informacija rukuje i s mogućim preusmjeravanjima. Preusmjeravanja se preuzimaju iz *Location* parametra HTTP odgovora ili iz *Meta* elementa HTML dokumenta. Ukoliko su zahtjevi unutar domene Web aplikacije , tada se i oni dodaju u listu neposjećenih linkova.

Postupak pretraživanja linkova traje relativno dugo, ipak određeno ubrzanje se postiže na sljedeće načine:

- ograničavanjem dubine pretraživanja,
- skraćivanjem maksimalnog vremena dohvata pojedinog HTML dokumenta i
- zabranjivanjem uzoraka linkova korištenjem pametnog modula.

Za vrijeme postupka pretraživanja linkova, modul za prikupljanje informacija sa svakog dohvaćenog dokumenta skuplja i forme. Informacije o svim pronađenim formama se zapisuju u listu formi, a zapis ima sljedeću strukturu:

- ime forme,
- metoda – POST ili GET,
- URL na kojem se forma nalazi,
- URL na koji se parametri forme proslijeđuju i
- popis svih polja forme, tj. imena, vrsta i vrijednosti polja.

4.1.3. Pametni modul

Smart modul ili pametni modul je integriran u modul za prikupljanje informacija kako bi značajno ubrzao istraživanje strukture linkova određene Web aplikacije. Za razliku od modula za prikupljanje informacija, pametni modul ne pamti vrijednosti linkova koji su pronađeni, nego linkove organizira u uzorke. Pri svom radu koristi dvije podatkovne strukture: listu uzoraka i rječnik parametara i vrijednosti.

Sljedećim primjerom, prikazana je organizacija linkova u uzorke :

```
URL: /index.php?id=3&option=com_task
URL: /index.php?id=1&option=com_news
URL: /index.php?id=3&option=com_content&task=view
```

Na temelju dohvaćenih linkova, pametni modul koristi svoje podatkovne strukture.

Lista uzoraka sadrži sljedeće zapise:

```
[[/index.php, id , option],2]
[[/index.php, id , option],1]
```

Riječnik parametara i vrijednosti sadrži sljedeće zapise:

```
{ id: [_INTEGER], option: [com_task, com_news, com_task], task:view }
```

Zapis u listi uzoraka sadrži relativni link do dokumenta (početni dio URL-a) te sve parametre koji su pronađeni u određenom linku. Svaki zapis sadrži i informaciju o broju pojavljivanja uzorka linka. U rječniku parametara su sadržani parametri svih linkova s svim pripadnim vrijednostima.

Ukoliko broj pojavljivanja uzorka pređe granicu određenu pragom pametnog modula, tada se će se uzorak zabraniti (dodati u listu zabranjenih linkova). Na taj način modul za prikupljanje informacija neće više dohvaćati HTML dokumente čiji se linkovi podudaraju s zabranjenim uzorkom.

Pametni modul je nužan za rad modula za otkrivanje vrste Web aplikacija. Kako se modul za otkrivanja vrste Web aplikacije koristi uzorcima linkova, potrebno je obaviti organizaciju linkova u uzorke linkova korištenjem pametnog modula. Također, pametnim modulom se može regulirati količina sadržaja potrebnih za određivanja vrste Web aplikacije, odnosno brzina prikupljanja informacija.

4.1.4. Implementacija modula za prikupljanje informacija

Programska implementacija modula za prikupljanje informacija je ostvarena korištenjem Python programskog jezika. Dijelovi kao što su metode za pretraživanje linkova, prikupljanje formi i parsiranje HTML dokumenata, su preuzeti od Wapiti-a. Modul za prikupljanje informacija je u potpunosti neovisan o drugim modulima unutar WSAT-a, pa se može i samostalno koristiti pri istraživanju Web aplikacija.

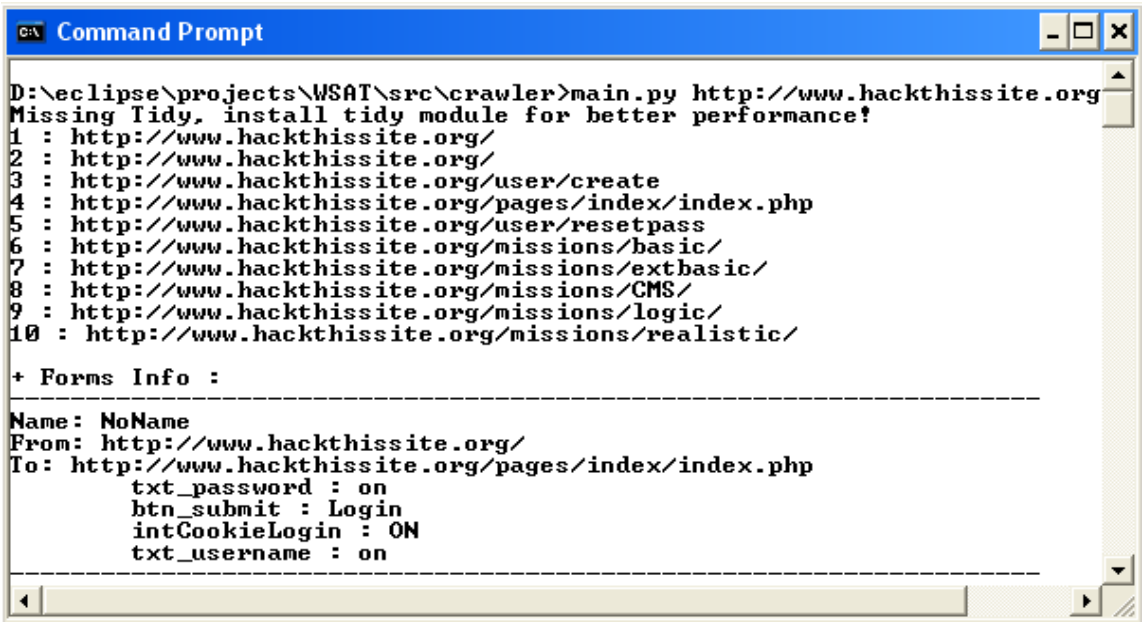
Modul za prikupljanje informacija se pokreće iz komandne linije s opcijama prikazanim u tablici 4.1:

Tablica 4.1. Opcije modula za prikupljanje informacija

Opcija	Duga opcija	Opis
-s <url>	*--start <url>*	*Specifikacija početnog URL-a.*
-x <url>	*--exclude <url>*	*URL koji se želi isključiti prilikom istraživanja, također se može koristiti i regularni izraz.*
-c <cookie_file>	*--cookie <cookie_file>*	*Korištenje sjedničkog indentifikatora .*
-r <parameter_name>	*--remove <parameter_name>*	*Uklanjanje određenih parametara iz URL-a.*
-d <depth>	*--depth <depth>*	*Korištenje dubine pretraživanja.*
-m <frequency>	*--smart <frequency>*	*Korištenje pametnog modula pri pretraživanju.*
-v <level>	*--verbose <level>*	*Definiranje ispisa: 0: Ispisuju se samo rezultati (na kraju rada) 1: Ispisuje se točka za svaki pronađeni URL (za vrijeme prolaska) 2: Ispisuje se svaki pronađeni URL (za vrijeme prolaska)*
-t <timeout>	*--timeout <timeout>*	*Definira se max. vrijeme čekanja za zahtjev (u sekundama)*
-h	*-help*	*Ispis pomoćne poruka.*

U sljedećem primjeru će se prikazati mogućnosti modula za prikupljanje podataka. Prvo se promatra istraživanje Web aplikacije bez korištenja sjedničkog identifikatora. Upisom sljedeće naredbe pokreće se modul za prikupljanje informacija te se ispisuju rezultati prikupljanje kao što je prikazano na slici 4.3.

```
>python main.py http://www.hackthissite.org
```



Slika 4.3. Prikaz rada modula za prikupljanje informacija bez korištenja sjedničkog identifikatora

Modul za prikupljanje informacija je prikupio prvih deset početnih URL-ova koji se pojavljuju na domeni za testiranje `http://www.hackthissite.org`. Također je pronašao i formu za prijavu s pripadnim informacijama i poljima.

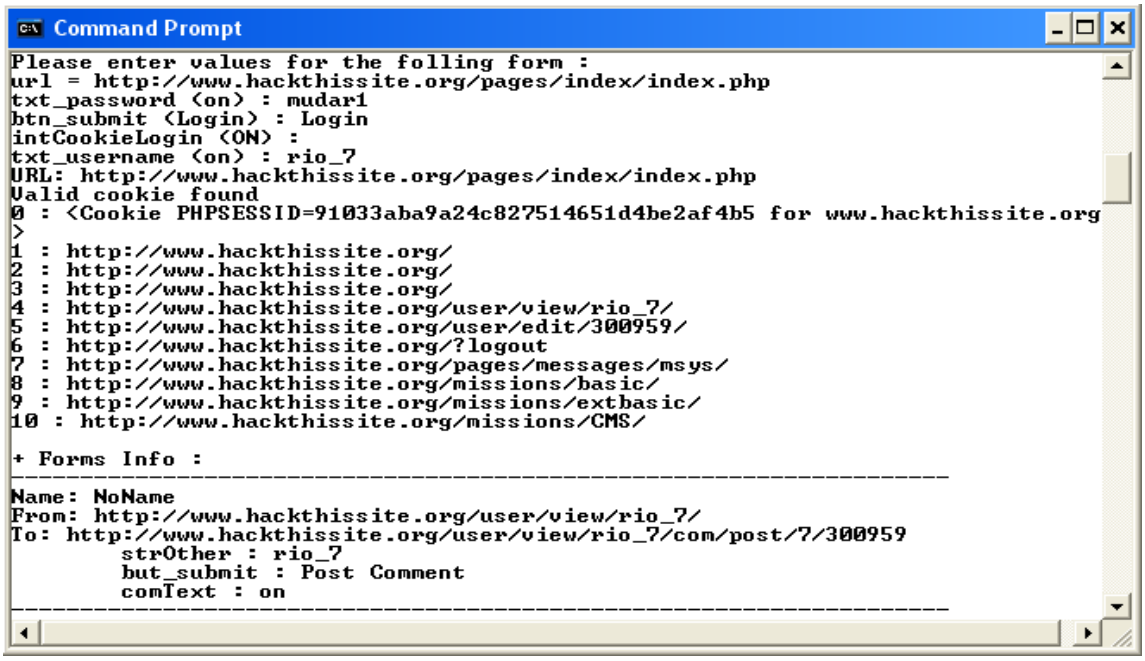
Istraživanje Web aplikacije korištenjem sjedničkog indentifikatora:

```
>python main.py http://www.hackthissite.org -c cookie.txt
No cookie file found,gathering new cookie:
```

Pošto datoteka s sjedničkim identifikatorom ne postoji, potrebno je ispuniti formu za prijavu:

```
Please enter values for the folling form :
url = http://www.hackthissite.org/pages/index/index.php
txt_password (on) : mudar1
btn_submit (Login) : Login
intCookieLogin (ON) : on
txt_username (on) : rio_7
```

Nakon ispunjavanja forme, provjerava se valjanost sjedničkog identifikatora, ispisuje se njegova vrijednost te modul za prikupljanje informacija započinje s istraživanjem kao što je prikazano na slici 4.4.



Slika 4.4. Prikaz rada modula za prikupljanje informacija korištenjem sjedničkog identifikatora

Vidljiva je razlika između URL-ova dobivenih istraživanjem bez sjedničkog identifikatora i dobivenih istraživanjem s sjedničkim identifikatorom, što ukazuje na različitu razinu autorizacijskih prava. Također je primjećeno da prvih pet URL-ova ne posjeduje formu za logiranje, što ukazuje da se radi o autoriziranom korisniku koji se uspješno prijavio na Web aplikaciju. Ali URL pod brojem `7 : http://www.hackthissite.org/pages/messages/msys/` posjeduje formu za logiranje. Razlog tomu je URL pod brojem `6 : http://www.hackthissite.org/?logout` kojim se korisnik odjavio iz sustava.

Ovaj primjer upućuje na problem pri automatiziranom istraživanju strukture Web aplikacije korištenjem autorizacijskih prava određenog korisnika. Problem je što automatizirani pretraživač ne prepoznaje stranicu za odjavu korisnika. Stoga je potrebna intervencija čovjeka pri ovoj vrsti istraživanja. Gornji problem se može izbjeći isključivanjem URL-a za odjavu:

```
>main.py http://www.hackthissite.org -c cookie.txt -x
http://www.hackthissite.org/?logout
```

4.2. Modul za otkrivanje vrste Web aplikacije

Mnogi automatizirani alati otvorenog koda pri procesu prikupljanja informacija otkrivaju vrstu Web poslužitelja, te vrstu i verziju operacijskog sustava na kojima se pokreće određena Web aplikacija. Međutim, ne postoje dovoljno razvijeni alati otvorenog koda specijalizirani za otkrivanje vrste i verzije Web aplikacije.

Proces otkrivanja vrste Web aplikacije predstavlja značajan korak u sigurnosnoj analizi Web aplikacije. Ukoliko se uspješno provede taj proces, odnosno ako se točno odredi o kojoj se Web aplikaciji radi, tada se značajno olakšava proces otkrivanja ranjivosti unutar Web aplikacije. Kako većina današnjih Web aplikacija otvorenog koda ima popriličan popis

ranjivosti koji je dostupan na Internetu, postupak kompromitiranja Web aplikacije se svodi na upućivanju dostupne napadačke sekvence putem jednog ili više HTTP zahtjeva prema određenoj Web aplikaciji.

Web aplikacije otvorenog koda, kao što su sustavi za upravljanje sadržajem (eng. Content Manager System – CMS), portali, *Web-Mail* aplikacije i druge Web aplikacije imaju predefiniranu strukturu linkova, predefinirani niz formi unutar određene HTML stranice, pa čak i predefinirani dio sadržaja unutar HTML dokumenata. Vrlo rijetko se administratori tih Web aplikacija zamaraju s izmjenom osnovnog koda Web aplikacije kako bi prikrili tragove koji bi mogli otkriti vrstu i verziju Web aplikacije. Eventualne izmjene se najčešće svode na promjenu zaglavlja HTML dokumenata u kojoj se prikazuje vrsta Web aplikacije, kako bi se uklonio najočitiji trag. Slika 4.5. prikazuje zaglavlje vrste Web aplikacije kojoj nije prikriven očiti trag.

Slika 4.5. Trag s imenom vrste Web aplikacije u zaglavlju

Cilj modula za otkrivanje vrste Web aplikacije je otkriti vrstu Web aplikacije kada nisu prisutni najočitiji tragovi. Automatizirani postupak otkrivanja nema velikog smisla ako se oslanja na vrstu informacija do kojih se lako dolazi i na temelju kojih se jednostavno otkriva vrsta Web aplikacije. Takav način otkrivanja je pogodniji za ljude. Pri određivanju vrste Web aplikacije, uzimaju se u obzir pouzdanosti pojedinih informacija. Informacije su pouzdanije ako ih je teže izmijeniti i eliminirati iz Web aplikacije, odnosno ako se mogu pronaći u većem broju Web aplikacija iste vrste. Što je vrsta informacije pouzdanija, to je njen utjecaj veći na odluku o vrsti Web aplikacije.

Polazeći od pretpostavke da se ne mijenja osnovni kod same Web aplikacije, pri procesu udaljenog otkrivanju Web aplikacije potrebno je obratiti pozornost na tri vrste korisnih informacija:

- ključne riječi,
- uzorci linkova i
- forme.

Ključne riječi su karakteristični znakovni nizovi koji se pojavljuju unutar HTML dokumenata određene Web aplikacije. Na primjer, ključne riječi za *Mambo* sustav za upravljanje sadržajem su sljedeće:

- *Mambo,*
- *Powered by Mambo i*
- *http://mambo-foundation.org.*

Ključne riječi nisu pouzdane informacije na temelju kojih se može odrediti vrsta Web aplikacije. Osim što se jednostavno eliminiraju iz HTML dokumenata Web aplikacije, Web aplikacije mogu u nekom svom kontekstu koristiti ključnu riječ sasvim druge vrste Web aplikacije. Primjerice, Web aplikacija čiji sadržaj opisuje latinoameričke plesove, može često koristiti riječ Mambo u nekom HTML dokumentu, a da pritom vrsta te Web aplikacije nema nikakve veze s Mambo sustavom za upravljanjem sadržajem (CMS). Ako se uzme u

obzir nepouzdanost ključnih riječi i činjenica da ključne riječi predstavljaju očiti trag pri otkrivanju vrste Web aplikacije, ključnim riječima se će dati minimalni značaj u postupku automatiziranog otkrivanja vrste Web aplikacije.

Uzorci linkova su sve moguće kombinacije URL-ova koje se mogu pojaviti na određenoj Web aplikaciji. Uzorak se sastoji od sljedećih komponenti:

1. Osnovnog URL-a koji sadrži ime domene, te put i naziv do određene datoteke.
2. Popis svih parametara koji su sadržani u uzorku.
3. Popis svih vrijednosti za svaki od parametara sadržanih u uzorku.

Svaka Web aplikacija posjeduje određeni skup uzoraka linkova. Linkovi u Web aplikaciji se mogu isključivo promijeniti izmjenama osnovnog koda Web aplikacije. Ako se uzme u obzir pretpostavka da se osnovni kod Web aplikacije ne mijenja, proizlazi da su uzorci linkova vrlo pouzdana vrsta informacija dostupna u Web aplikaciji. Drugim riječima, uzorci se vrlo rijetko razlikuju u više Web aplikacije iste vrste. Stoga se prikupljanjem i međusobnim uspoređivanjem uzoraka Web aplikacija, može se prilično točno otkriti vrsta određene Web aplikacije. Kako bi se umanjila mogućnost krive procjene vrste i verzije Web aplikacije, potrebno je imati velik skup uzoraka za relativno velik broj različitih Web aplikacija. Čest je slučaj, da dvije različite Web aplikacije imaju sličan skup uzoraka, pa je potrebna pomoć pri donošenju odluka na temelju drugih korisnih informacija.

Forme su ulazne točke pomoću kojih korisnik predaje podatke Web aplikaciji. Forma sadrži veći broj ulaznih elemenata (engl. *Input*) koji se razlikuju po funkcionalnosti i prikazu ulaznih podataka. Primjerice, forma može posjedovati obična polja za upis teksta, polja za izbor dvaju ili više vrijednost (engl. *Checkbox)* ili polja za izbor jedne vrijednosti (engl. *RadioButton*). U HTML dokumentu svi ulazni elementi se nalaze unutar <*form*> elementa koji definira formu. Uz ulazne elemente, forma obično sadrži ime, identifikator, metodu slanja i URL na koji se proslijeđuju vrijednosti upisane od strane korisnika. Web aplikacije često imaju karakteristične forme unutar svojih HTML dokumenata koje se mogu jednostavno identificirati. Identifikacija formi se temelji na usporedbi imena formi, usporedbi naziva ulaznih elemenata, te tipova i vrijednosti ulaznih elemenata. Na primjer, forma za prijavu se identificira usporedbom imena forme koji najčešće počinje s *login* i najčešće sadrži dva polja za unos teksta s pripadnim nazivom i vrijednosti koji se razlikuju od aplikacije do aplikacije.

Problem s formama je što Web aplikacije iz praktičnih razloga ne koriste ponuđen skup formi, već koriste samo najbitnije forme za prijavu ili traženje sadržaja po Web aplikaciji. Forme se po potrebi, mogu jednostavno eliminirati iz Web aplikacije korištenjem administratorskog izbornika. S druge strane, ukoliko je forma prisutna na Web aplikaciji, njena struktura se može promijeniti samo izmjenama osnovnog koda Web aplikacije. Iz svega navedenog, pretpostavlja se da su forme manje pouzdane informacije od uzoraka linkova, a više pouzdane od ključnih riječi.

Modul za otkrivanje vrste Web aplikacije može se koristiti kao:

- generator fingerprint XML datoteke i
- modul za usporedbu i određivanje vrste Web aplikacije.

Neovisno o načinu rada, modul za otkrivanje vrste i verzije Web aplikacije koristi modul za prikupljanje informacija kako bi prikupio korisne informacije o Web aplikaciji. Generator fingerprint XML datoteke strukturira i zapisuje korisne informacije u XML datoteku koju WSAT koristi u budućnosti pri utvrđivanju vrste i verzije Web aplikacije. Modul za usporedbu i utvrđivanje vrste i verzije Web aplikacije, na temelju gotovih *fingerprint* XML datoteka, uspoređuje prikupljene informacije s onima iz XML datoteke kako bi utvrdio vrstu i verziju Web aplikacije.

4.2.1. Generiranje *fingerprint* XML datoteke

Modul za otkrivanje vrste Web aplikacije posjeduje generator koji prikupljene informacije zapisuje u fingerprint XML datoteku. Struktura *fingeprint* XML datoteke (bez podataka) je prikazana na slici 4.6.

```
<?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 4.6. Struktura fingerprint XML datoteke

Početna dva elementa XML dokumenta su <*name*> i <*id*> elementi, koji predstavljaju ime i identifikator Web aplikacije. Imenom se opisuje vrsta i verzija određene Web aplikacije, dok je identifikator jedinstveni broj pridružen određenoj vrsti i verziji Web aplikacije. Ime i identifikator zadaje korisnik WSAT-a prije generiranja datoteke.

Korisnik također upisuje i ključne riječi određene Web aplikacije. Ključne riječi se zapisuju u <*keywords*> element u kojem svaka ključna riječ ima svoj podelement <*keyword*>. Slika 4.7 prikazuje dio XML datoteke u koji se upisuje ime, identifikator *i ključne riječi:*

```
<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 4.7. Zapis imena, identifikatora i ključnih riječi u XML datoteci

Nakon unošenja korisničkih podataka, započinje proces prikupljanja informacija pomoću modula za prikupljanje informacija. Kako proces prikupljanja informacija traje relativno dugo modul za prikupljanje informacija pri svom radu koristi pametni modul. Kao što je već rečeno, pametnom modulu se zadaje određeni prag kojim se može definirati brzina i potpunost pretraživanja. Ukoliko je prag pametnog modula mali, tada će pretraživanje trajati znatnokraće, ali će i količina prikupljenih korisnih informacija biti mala. S druge strane, ukoliko je prag veći, tada će pretraživanje trajati znatno dulje, ali će i količina prikupljenih korisnih informacija biti znatno veća. Pri procesu generiranja potrebno je koristiti što veći prag pametnog modula kako bi dobili što veći broj uzoraka i formi, odnosno kako bi se prikupilo što više korisnih informacija koji će olakšati procjenu vrste i verzije Web aplikacije.

Uzorci linkova, prikupljeni od strane modula za prikupljanje informacija se zapisuju u obliku prikazanom slikom 4.8.

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

Slika 4.8. Zapis uzoraka linkova u XML datoteci

Svi uzorci linkova se upisuju u element <*link_patterns*> koji sadrži element <*separator*> i popis pojedinog uzorka unutar <*pattern*> elementa.

U element <*separator*> se upisuje znak koji razdvaja parametre URL uzoraka. Današnje Web aplikacije obično koriste znak „&“ u razdvajanju parametara, ali sve se češće javljaju i drugi složeniji znakovi. Primjerice, koriste se nizovi znakova „QQ“ i „WW“ ili znak „/“ čije korištenje navodi na pomisao da Web aplikacija sadrži mnoštvo mapa, iako to nije slučaj. Sofisticirani znakovi znatno otežavaju analizu URL-ova, te automatizirano izdvajanje parametara i njihovih vrijednosti.

Pojedini uzorak unutar <*pattern*> elementa sadrži podelemente prikazane u tablici 4.2.

Tablica 4.2. Sadržaj elementa <pattern>

Podelement	Opis
<url>	*relativni link do pojedinog dokumenta*
<weight>	*broj pojavljivanja određenog uzorka, tj. koliko je puta uzorak pronađen za vrijeme prikupljanja informacija*
<params>	*popis svih prikupljeni imena parametara i njihovih vrijednosti određenog uzorka. Svaki parameter je zapisan u poseban element <param> koji sadrži atribut name, a sadržaj elemementa je popis svih vrijednosti koje su razdvojene znakom „|“*

Prikupljene forme se zapisuju u obliku prikazanom na slici 4.9.

```
<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 4.9. Zapis formi u XML datotoeci

Forme se upisuju u element <*forms*> koji sadrži popis svih formi. Pojedina forma se opisuje <*form*> elementom s atributima koji opisuju ime forme i relativnu lokaciju na kojoj se forma nalazi. Element <*form*> sadrži podelemente prikazane u tablici 4.3.

Tablica 4.3. : Sadržaje elementa <form>

Podelement	Opis
<from>	*URL na kojem se forma nalazi*
<to>	*URL na koji se prosljeđuju parametri forme*
<field>	*Ulazni element forme s atributom imena. Sadržaj elementa je prikupljena vrijednost ulaznog elementa.*

U sljedećem primjeru će se prikazati generiranje fingerprint XML datoteke. Za generiranje koristimo jednostavan program u Pythonu koji istancira klasu `generator` kojoj se zadaje URL Web aplikacije za koju se želi generirati *fingerprint* XML datoteka, prag pametnog modula i ime datoteke u koji se upisuju rezultati. Korisnik također postavlja vrijednost imena, identifikatora te ključnih riječi, kao što je prikazano u sljedećem odsječku:

```
GN=generator('http://test5.privatna.org' , 40,'/fingerprint.xml')
GN.setName('Drupal')
GN.setID(5)
GN.setKeywords(['Drupal','www.drupal.com'])
GN.writeXML()
```

Metodom writeXML() započinje prikupljanje podataka i upisivanje korisnih informacija u datoteku. Dobivena *fingerprint* XML datoteka je priložena u dodatku B.

4.2.1. Određivanje vrste Web aplikacije

Postupak određivanja vrste Web aplikacije se temelji na ocjeni podudarnosti korisnih informacija iz *fingerprint* XML datoteka s prikupljenim korisnim informacijama Web aplikacije kojoj se želi otkriti vrsta i verzija .Ukoliko se korisne informacije jedne vrste Web aplikacije zapišu u strukturiranu *fingerprint* XML datoteku, tada se na temelju usporedbe tih informacija s korisnim informacijama druge Web aplikacije može ustanoviti u kojoj su mjeri Web aplikacije slične.

U određivanju vrste i verzije Web aplikacija najvažnije je imati kvalitetnu *fingerprint* XML datoteku. Kvalitetna *fingerprint* XML datoteka sadrži veliki broj različitih uzoraka linkova i većinu formi koje se mogu naći u određenoj vrsti Web aplikacije. Velikim brojem uzoraka i formi se postiže veća podudarnost pri usporedbi ukoliko se radi o dvije Web aplikacije iste vrste, te je manja vjerojatnost pogrešnog određivanja vrste Web aplikacije.

Postupak određivanja započinje zadavanjem domene Web aplikacije kojoj želimo odrediti vrstu. Postupak se može podijeliti u 4. koraka:

1. Usporedba ključnih riječi
2. Usporedba uzoraka linkova
3. Usporedba formi
4. Konačna odluka

Usporedba ključnih riječi je početni korak u određivanju vrste Web aplikacije. Za usporedbu se koristi početna stranica Web aplikacije, odnosno njezin HTML sadržaj. Prolaskom kroz različite *fingerprint* XML datoteke računa se frekvencija pojavljivanja ključnih riječi unutar početne stranice Web aplikacije. *Fingerprint* XML datoteka koja posjeduje najveću frekvenciju pojave ključnih riječi se vrednuje s faktorom 1. Vrednovanje ostalih se vrši dijeljenjem vlastite frekvencije s najvećom frekvencijom pojave ključnih riječi.

Sljedećim primjerom se opisuje ocjenjivanje podudarnosti tri uzorka temeljem usporedbe s vrijednostima u listi prikupljenih uzoraka i rječnikom vrijednosti parametara.

Dvije *fingeprint* XML datoteke s pripadnim ključnim riječima su prikazane slikom 4.10.

```
<keywords>
      <keyword>Joomla!</keyword>
      <keyword>Powered by</keyword>
      <keyword>http://www.joomla.org</keyword>
</keywords>
<keywords>
            <keyword>Powered by</keyword>
            <keyword>Mambo</keyword>
            <keyword>http://mambo-foundation.org</keyword>
</keywords>
```

Slika 4.10. Dio Joomla i Mambo XML datoteke

Prva *fingeprint* XML datoteka odgovara Joomla CMS Web aplikaciji, a druga Mambo CMS Web aplikaciji. Korištenjem popisa ključnih riječi iz ovih datoteka u ispitivanju nemodificirane Joomla! Web aplikacije, Joomla bi imala frekvenciju pojavljivanja 3, a Mambo 1. Prema tome ocjena podudarnosti za Joomlu bi bila 1 ili 100%, a za Mambo 0.33 ili 33%. Ova metoda relativnog ocjenjivanja se koristi i u ostalim usporedbama.

Usporedba uzoraka linkova započinje prikupljanjem linkova nepoznate Web aplikacije. Modul za prikupljanje informacija prolazi kroz sve dostupne linkove, te uz korištenje pametnog modula, zapisuje sve uzorke linkova i vrijednosti parametara u pripadne liste. Pri usporedbi uzoraka linkova, koristi se lista uzoraka i lista vrijednosti parametara. Prolaskom kroj različite fingerprint XML datoteke, uspoređuju se uzorci linkova iz datoteke s prikupljenim uzorcima sadržanima u listi uzoraka. Ocjena podudarnosti uzoraka se provodi na sljedeći način:

1. Provjeravaju se relativni linkovi do dokumenta. Ukoliko se relativni link do dokumenta u polju <*url*> unutar uzorka XML datoteke razlikuje od relativnog uzorka unutar liste uzorka, tada uzorci nisu podudarni. Ako su relativni linkovi identični, prelazi se na točku 2.
2. Provjeravaju se imena parametara. Svaki parametar iz pojedinog uzorka XML datoteke se mora nalaziti u uzorku unutar liste uzoraka da bi uzorci bili podudarni. Parametri se moraju poklapati i u broju i u imenu.

Vrijednosti parametara se ne uzimaju kao bitan faktor u ocjeni podudarnosti uzoraka. Razlog tomu je što većina Web aplikacija dozvoljava definiranje vlastitih vrijednosti parametara, ili se oslanja na cjelobrojne ili alfanumeričke vrijednosti parametara koje nije jednostavno uspoređivati. Podudarnost vrijednosti parametara se određuje prilikom usporedbe parametara pri čemu se provjerava da li se vrijednost parametara iz XML datoteke nalazi u rječniku vrijednosti parametara. Ocjena podudarnosti vrijednosti je korisna u slučaju da se pri usporedbi uzoraka dobiju približno jednaki rezultati, pri čemu podudarnost vrijednosti može biti presudni faktor. Kod usporedbe uzoraka linkova se također koristi relativno ocjenjivanje. Dakle, *fingerprint* XML datoteka s najvećim brojem podudarnih uzoraka ima podudarnost 1 ili 100%, a ostale datoteke se skaliraju po toj vrijednosti.

Sljedeći primjer opisuje usporedbu i određivanje ocjena podudarnosti tri uzorka s prikupljenim informacijama u listi uzoraka i rječniku vrijednosti parametara.

Joomla uzorak je prikazan na slici 4.11 :

```
<pattern>
      <url>/index.php</url>
      <params>
      <param name="Itemid">_INTEGERS_</param>
      <param name="option">com_content|com_newsfeeds|com_search</param>
      <param name="task">view|lostPassword|register</param>
      </params>
</pattern>
```

Slika 4.11. Joomla uzorak

Mambo uzorak je prikazan na slici 4.12.

```
 <pattern>
      <url>/index.php</url>
      <params>
      <param name="Itemid">_INTEGERS_</param>
      <param name="option">com_contact|com_weblinks|com_content
      </param>
      </params>
</pattern>
```

Slika 4.12. Mambo uzorak

PHP-Nuke uzorak je prikazan na slici 4.13:

```
 <pattern >
<url>/modules.php</url>
      <params>
      <param name="name">Your_Account|Downloads|Submit_News</param>
      <param name="op">new_user  results</param>
      </params>
</pattern>
```

Slika 4.13. PHP-Nuke uzorak

Lista uzoraka i rječnik vrijednosti parametara ima sljedeći sadržaj:

```
[[/index.php, Itemid , option, task],2]]
{ id: [_INTEGER], option : [com_content,com_search], task : view}
```

U prvom koraku usporedbe, vrši se provjera relativnih linkova uzoraka. Joomla i Mambo u svom uzorku sadrže identičan relativni link *index.php* onomu u listi, pa se prelazi na sljedeći korak usporedbe. PHP-Nuke sadrži različit relativni link *modules.php* od *index.php*, pa je njegov uzorak nepodudaran. U sljedećem koraku, dohvaća se jedan po jedan parametar iz preostala dva XML uzorka. Joomla uzorak sadrži ispravan broj parametara i imena njegovim parametara se poklapaju s onima u listi, čime se njegov XML uzorak ocjenjuje podudarnim. Podudarnost vrijednosti parametara je 4 jer četiri vrijednosti parametara iz XML uzorka odgovaraju vrijednostima u listi. Mambo XML uzorak sadrži dva parametra koji odgovaraju po imenima, ali ne sadrži treći parametar *task*. Stoga, njegov XML uzorak nije podudaran s onim u listi uzoraka. Za ovaj uzorak, podudarnost vrijednosti parametara je 2.

Za usporedbu formi, koristi se lista formi dobivena u procesu prikupljanja informacija. Prolaskom kroz različite *fingerprint* XML datoteke uspoređuju su imena formi i imena polja formi iz datoteka s onima u listi prikupljenih formi. Ocjena podudarnosti formi se provodi na sljedeći način:

1. Ako je ime forme u XML datoteci identično imenu forme pronađene na stranici Web aplikacije (prikupljena lista formi), tada se prelazi na drugi korak usporedbe. U suprotnom, forme nisu podudarne, pa se prelazi na usporedbu iste forme iz XML datoteke s sljedećom formom pronađenom na stranici Web aplikacije.
2. Ako određeni postotak imena polja formi iz XML datoteke odgovara onima u listi formi, tada su forme podudarne. Postotak podudarnosti polja se određuje od strane korisnika.

Drugi uvjet podudarnosti formi ostavlja prostor za fleksibilnu usporedbu formi. Naime, kod nekih Web aplikacija ubacuju se nazivi imena polja u ovisnosti o korisničkim podacima ili sjedničkim identifikatorima. Ako se za takvu Web aplikaciju izgenerira *fingerprint* XML datoteka koja će se koristiti za ispitivanje drugih aplikacija, vrlo je vjerojatno da će se pri usporedbi istih vrsta Web aplikacija pojaviti forme s istim imenima, ali će se imena određenih polja razlikovati, iako se radi o istim formama. Stoga se umjesto apsolutne 100%-ne podudarnosti, uvodi fleksibilnija ocjena podudarnosti. Vrijednosti polja formi se ne uzimaju u obzir pri ocjeni podudarnosti formi. Vrijednosti polja su podložne promjenama od strane administatora Web aplikacije i ovise o lokalizaciji Web aplikacije, pa postoji mogućnost da će polje forme imati različite vrijednosti za isto ime polja u određenoj vrsti Web aplikacije. Slika 4.14 prikazuje formu s dinamički dodanim poljem.

```
<form name="login" location="/">
     <from>/</from>
     <to>/index.php</to>
     <field name="username">on</field>
     <field name="lang">english</field>
     <field name="return">http://www.lexnetcg.com/</field>
     <field name="remember">yes</field>
     <field name="op2">login</field>
     <field name="passwd">on</field>
     <field name="j11bc67039af2dbc8bf8da3755131cb2d">1</field>
     <field name="Submit">Login</field>
     <field name="option">login</field>
</form>
```

Slika 4.14. Forma s dinamički dodanim poljem

Polje s imenom *j11bc67039af2dbc8bf8da3755131cb2d* je dinamički dodano polje u koje Web aplikacija zapisuje sjednički identifikator korisnika. Prema tome, sadržaj forme i imena polja isključivo ovise o korisniku.

Na ocjenu podudarnosti formi ne utječe ni lokacija formi, odnosno URL na kojem se forma nalazi. Razlog tomu je što se forme u pravilu ponavljaju na više lokacija, pa nema smisla uspoređivati iste forme koje se nalaze na različitim lokacijama. Primjer formi koje se nalaze na više lokacija unutar Web aplikacije su forme za prijavu i forme za pretraživanje. One su u najkorisnije i može ih se naći u većini Web aplikacija, pa je nužno da fingerprint XML datoteka sadrži zapis o formi za prijavu i pretraživanje.

Nakon utvrđivanja podudarnosti korisnih informacija, donosi se konačna odluka o vrsti Web aplikacije. Konačna odluka se donosi na temelju ukupne ocjene podudarnosti koja se računa sljedećom formulom:

UKUPNA_OCJENA = k * op_ključne_riječi +u * op_uzorci_linkova + f * op_forme

pri čemu **k + u + f = 1**

U formuli, **k** je težišni faktor ključnih riječi, **u** težišni faktor uzoraka linkova i **f** težišni faktor formi. Težišni faktori se množe s pripadnim ocjenama podudarnosti koje su prethodno relativno ocjenjene, odnosno skalirane na najbolju pripadnu ocjenu podudarnosti. Ako se uzme u obzir pretpostavka da su uzorci linkova pouzdaniji faktor od formi, a forme od ključnih riječi, tada se pri računanju ukupne ocjene podudarnosti poštuje sljedeći odnos **u > f > k**. Optimalne vrijednosti težišnih faktora će se utvrditi pri ispitivanju sustava u sljedećem poglavlju.

Web aplikacija koja ima ukupnu ocjenu podudarnosti 1 ili 100% pokazuje najbolje rezultate po sva tri kriterija. Ukoliko određena vrsta Web aplikacije ima ukupnu ocjenu 100%, tada se može poprilično sigurno reći da je otkrivanje vrste Web aplikacije ispravno. S druge strane, Web aplikacije s ukupnom ocjenom podudarnosti < 100% ostavljaju određenu nedoumicu, pa i konačna odluka nije definitivna. U tom slučaju je potrebno razmotriti nekoliko vrsta Web aplikacija s najboljom ukupnom ocjenom podudarnosti. Ispravnim otkrivanjem vrste Web aplikacije značajno se olakšava postupak otkrivanja potencijalnih ranjivosti, pošto se samo ispituje prisutnost ranjivosti koji su tipični za određenu vrstu Web aplikacije.

4.3. Modul za otkrivanje ranjivosti

Kako većina Web aplikacija posjeduje određeni broj poznatih ranjivosti, otkrivanjem vrste Web aplikacije se u biti otkrivaju potencijalne sigurnosne rupe. Izmjenama u osnovnom kodu ili korištenjem sigurnosnim zakrpa, stručnjaci za sigurnost ili administratori Web aplikacija mogu na vrijeme otkloniti potencijalne sigurnosne rupe. Stoga je potrebno pažljivo provesti postupak otkrivanja ranjivosti kako bi se s sigurnošću utvrdilo postojanje određene ranjivosti u Web aplikaciji.

Modul za otkrivanje ranjivosti koristi automatizirani postupak otkrivanja ranjivosti. Nakon određivanja vrste Web aplikacije, modulu su dostupne sve potencijalne ranjivosti za tu vrstu Web aplikacije. Ranjivosti su opisane XML datotekama, te su prilagođene postupku otkrivanja ranjivosti. Sam postupak se sastoji od dvije faze:

- faza napada,
- faza analize.

Faza napada se sastoji od točno određenog slijeda koraka kojima se želi kompromitirati Web aplikacija. Svaki korak je u biti jedan HTTP zahtjev koji sadrži metodu slanja (POST ili GET), URL, te sadržaj zahtjeva prilagođen određenoj vrsti ranjivosti. Slika 4.15 prikazuje dio XML datoteke u kojoj se opisuje faza napada za SQL ranjivost.

Vidljivo je, da se faza napada sastoji od jednog koraka, točnije od jednog HTTP POST zahtjeva s pripadnim podacima. Među podacima koji se šalju prema Web aplikaciji na određeni URL su parametri i njihove pripadne vrijednosti. Posebno je zanimljiv parametar

```
<Exploit>
 <Step method="post" url="/modules.php?name=Encyclopedia&file=search">

      <Postdata>query=m &amp eid=1'/**/UNION SELECT pwd as title FROM
                              nuke_authors WHERE '1'='1
      </Postdata>
 </Step>
</Exploit>
```

Slika 4.15. Zapis faze napada u XML datoteci

eid, kojem se iza vrijednosti 1, umeće znakovni niz tipičan za SQL ubacivanje. Cilj napada je SQL ubacivanjem izvući sve lozinke korisnika Web aplikacije.

U fazi analize se utvrđuje da li je faza napada uspješno izvršena, odnosno da li Web aplikacija zaista posjeduje poznatu ranjivost. Faza analize ovisi vrsti ranjivosti, a najčešća metoda je usporedba HTML dokumenta dobivenih za različite vrijednosti parametara.

Slika 4.16 prikazuje dio XML datoteke u kojoj se opisuje faza analize za SQL ranjivost:

```
<Analyze>
 <Step method="post" url="/modules.php?name=Encyclopedia&file=search">
      <Postdata>query=m &amp eid=1
      </Postdata>
 </Step>
</Analyze>
```

Slika 4.16. Zapis faze analize u XML datoteci

Vidljivo je, da je faza analize slična gornjoj fazi napada. Razlika je samo u vrijednosti parametra *eid*. U fazi analize koja se temelji na usporedbi HTML dokumenata, dohvaćaju se i uspoređuju HTTP dokumenti s "napadačkim" i "normalnim" vrijednostima parametara. Ukoliko se dokumenti bitno razlikuju, odnosno ukoliko se otkriju informacije koje ne bi smjele biti otkrivene, tada ranjivost zaista postoji.

Automatizirani postupak otkrivanja ranjivosti je prilagođen otkrivanju tehničkih ranjivosti, dok su za logičke ranjivosti potrebne složenije radnje koje uključuju i ljudsku intervenciju. U pravilu, svaka vrsta tehničkih ranjivosti ima različitu metodu napada i analize pomoću kojih se vrši automatizirano otkrivanje određene tehničke ranjivosti. Postupci automatizacije pojedinih vrsta ranjivosti:

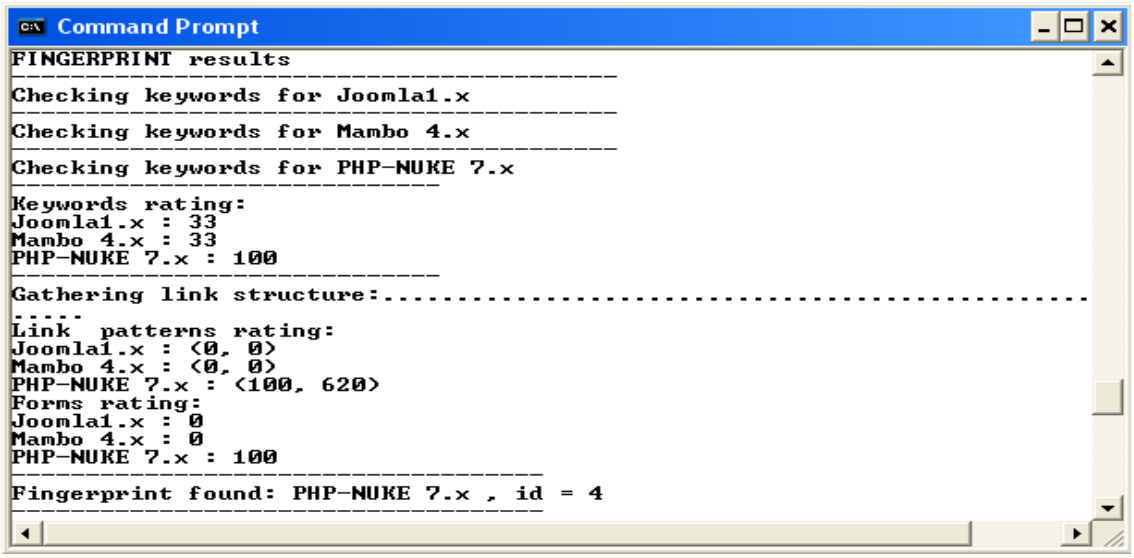
- Za napade grubom silom i autentifikacijske/autorizacijske napade već postoje gotovi automatizirani napadački alati pomoću kojih se mogu odrediti korisničke lozinke i druge tajne informacije, te poduzeti kompromitirajuće radnje vezane uz sjedničke identifikatore. Faza napada se poklapa s radom alata. U fazi analize je potrebno odrediti valjanost sjedničkog identifikatora, bilo ispunjavanjem forme s korisničkim podacima, korištenjem ukradenog ili namještenog sjedničkog identifikatora.
- XSS napad, odnosno njegova faza napada se ostvaruje umetanjem kompromitirajuće skripte u neki od ranjivih parametara. Aktivnost skripte se u pravilu ne može očitati iz HTML dokumenta. Stoga je bitno je da skripta ostavi određeni trag kao što je zapis u datoteteku ili bilo kakvu povratna informacija, kako bi se u fazi analize mogla provjeriti uspješnost napada. Ako trag postoji, tada je skripta uspješno izvršena i XSS ranjivost je prisutna.

- SQL ubacivanje se ostvaruje umetanjem specijalnih znakova i SQL naredbi u vrijednost nekog od ranjivih parametara kako bi se iz baze dohvatile tajne informacije ili kompromitirala baza podataka. Faza analize se temelji na usporedbi HTML dokumenata, pri čemu se uspoređuju pravilni HTML dokumenti s potencijalno ranjivim HTML dokumentima (parametrima s SQL ubacivanjem). Ukoliko postoji značajna razlika između HTML dokumenata, tada je SQL ranjivost prisutna.
- Napadi vezani uz izvršavanje naredbi se također provode umetanjem određenih OS naredbi ili kombinacije znakova u vrijednost parametara. Faza analize je slična onoj kod XSS naredbe. Napad mora ostaviti trag ili poslati povratnu informaciju kako bi se utvrdila prisutnost ranjivosti.
- Otkrivanje informacije se provodi zadavanjem različitih vrijednosti URL-a koji se sastoje od specijalnih znakova i putova do određenih datoteka. U fazi analize je potrebno provjeriti dostupnost pojedinog resursa. Dostupnost se može jednostavno provjeriti statusnim kodom (200 Ok) ili usporedbom HTML dokumenta, ukoliko se radi o izlistavanju mapa.
- DoS napadi, iako spadaju u grupu logičkih ranjivosti, se donekle mogu automatizirati. Neovisno o vrsti DoS napada i fazi napada, faza analize se provodi provjerom dostupnosti Web aplikacije i rada njenih usluga. Stoga je potrebno pažljivo provjeravati prisutnost DoS ranjivosti, kako se pri automatiziranom postupku ne bi značajnije ugrozio rad Web aplikacije.

Potrebno je napomenuti da WSAT-ov modul za otkrivanje ranjivosti posjeduje mogućnost automatizirano otkrivanja samo poznatih SQL ranjivosti za određenu vrstu Web aplikacije. Programska implementacija automatiziranih postupaka otkrivanja drugih vrsta ranjivosti premašuje okvir ovog diplomskog rada.

U sljedećem primjeru se opisuje otkrivanje ranjivosti korištenjem WSAT-a za PHP-Nuke 7.9 Web aplikaciju. Cilj primjera je prikazati rad WSAT-a pri otkrivanju ranjivosti vezane uz SQL ubacivanje.

Rad WSAT-a započinje zadavanjem početnog URL-a ili domene Web aplikacije koju se želi ispitati. Modul za otkrivanje ranjivosti predaje modulu za otkrivanje vrste Web aplikacije početni URL kako bi obavio potrebno određivanje vrste Web aplikacije. Određivanje vrste je prikazano na slici 4.17.

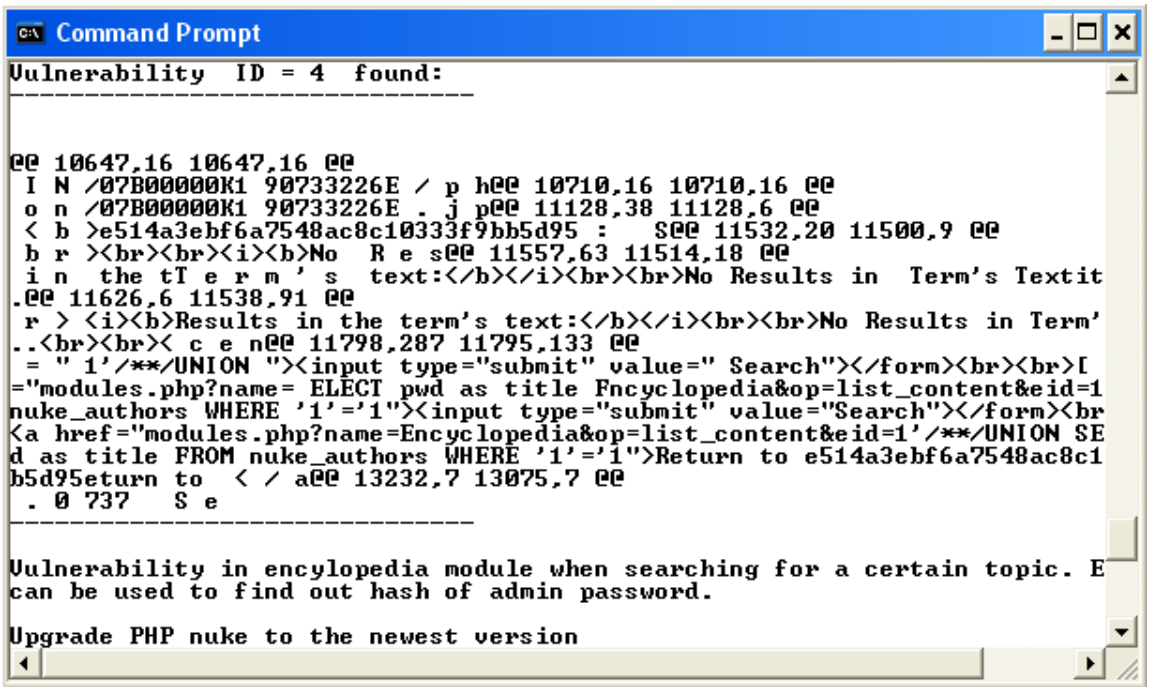


Slika 4.17. Rad modula za otkrivanje vrste Web aplikacije

Modul za otkrivanje vrste Web aplikacije posjeduje tri fingerprint XML datoteke na temelju kojih donosi konačnu odluku:

- PHP-Nuke 7.x,
- Mambo 4.x i
- Joomla 1.x

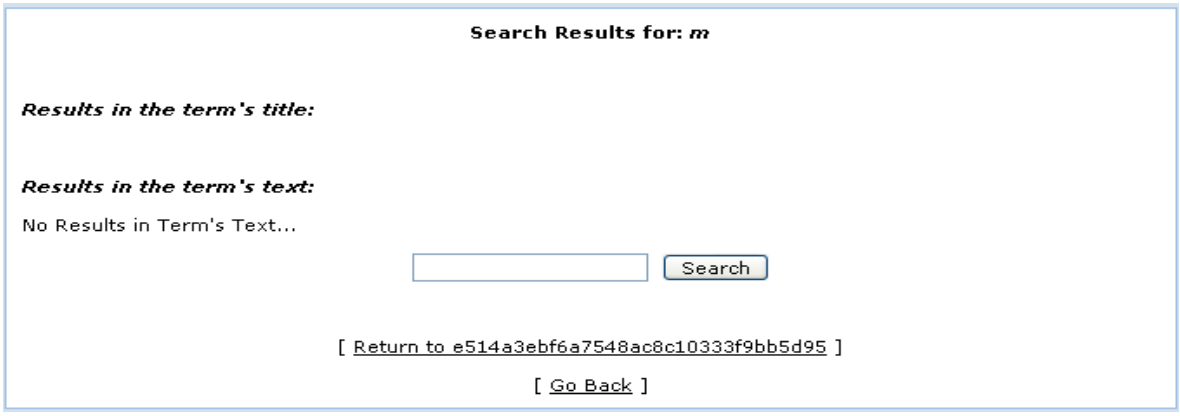
Prvotno se ocjenjuje podudarnost ključnih riječi. Vidljivo je, da PHP-Nuke 7.x ima najbolju ocjenu podudarnosti od 100%, dok Mambo 4.x i Joomla 1.x imaju 50%. Nakon ocjenjivanja podudarnosti ključnih riječi, počinje faza prikupljanja informacija. U toj fazi, modul za prikupljanje informacija u ovisnosti u pragu pametnog modula prikuplja sve uzorke linkova i forme. Na temelju prikupljenih informacija modul za određivanje vrste Web aplikacije ocjenjuje podudarnost uzoraka linkova i formi s uzorcima i formama iz pripadne *fingerprint* XML datoteke pojedine Web aplikacije. Ocjenjivanjem podudarnosti uzoraka linkova je utvrđeno da PHP-Nuke 7.x ima 100% ocjenu podudarnosti, dok Joomla i Mambo imaju 0%. Razlog tomu je što se temeljno, po relativnim linkovima, razlikuju uzorci linkova između PHP-Nuke 7.x, Joomla 1.x i Mambo 4.x. Čak i po ocjeni podudarnosti formi, PHP-Nuke 7.x ima maksimalni rezultat od 100%, dok Mambo 4.x ima 33% te Joomla 1.x 0%. Na temelju ocjena podudarnosti ključnih riječi, uzoraka linkova i formi, modul za otkrivanje vrste Web aplikacije donosi konačnu odluku prema kojoj ispitana Web aplikacija najbolje odgovara PHP–Nuke 7.x Web aplikaciji. Stoga, predaje se identifikator PHP-Nuke 7.x Web aplikacije modulu za otkrivanje ranjivosti i izvršava se postupak otkrivanja ranjivosti, kao što je prikazano na slici 4.18.



Slika 4.18. Otkrivanje ranjivosti korištenjem modula za otkrivanje ranjivosti

Na temelju identifikatora Web aplikacije, modul za otkrivanje ranjivosti pregledava svoju bazu ranjivosti kako bi izvukao sve ranjivosti vezane u PHP-Nuke 7.x Web aplikaciju. Za svaku ranjivost provodi se faza napada i faza analize. U ovom specijalnom slučaju, faza napada i faza analize odgovaraju gornjim primjerima. Analizom HTML dokumenata i njihovom usporedbom, utvrđuje se postojanje ranjivosti. Dodatno, prikazuje se nepodudarni dio HTML dokumenata u kojem se prikazuju dodatne informacije dobivene uspješnim SQL ubacivanjem. Prikaz iz Web preglednika otkriva razliku između HTML dokumenata (slika 4.19 i slika 4.20).

Slika 4.19. Normalni HTML dokument



Slika 4.20. HTML dokument dobiven SQL ubacivanjem

Iz gornjih slika i tekstualne razlike nepodudarnosti HTML dokumenata se otkriva sljedeći znakovni niz: *e514a3ebf6a7548ac8c10333f9bb5d95*. Ako se pogleda faza napada, prema Web aplikaciji se upućuje sljedeća SQL naredba:

```
/**/UNION SELECT pwd as title
           FROM  nuke_authors
           WHERE '1'='1
```

Ovom SQL naredbom se želi izvući sažetak administratorske lozinke. Upravo znakovni niz *e514a3ebf6a7548ac8c10333f9bb5d95* odgovara sažetku administratorske lozinke, pa se utvrđuje postojanje ranjivosti u enciklopedijskom modulu unutar PHP-Nuke 7.9 Web aplikacije. Da bi se pokrila ova sigurnosna rupa potrebno je nadograditi PHP – Nuke 7.9 na noviju verziju koja sadrži sigurnosnu zakrpu.

5. Ispitivanje sustava za otkrivanje vrste Web aplikacije

Cilj ispitivanja sustava za otkrivanje vrste Web aplikacije je određivanje pravilnih vrijednosti težišnih faktora na temelju kojih se donosi konačna odluka o vrsti Web aplikacije. Drugim riječima, potrebno je potvrditi pretpostavku da se uzorci linkova, forme i ključne mogu koristiti za određivanje vrste Web aplikacije i odrediti u kojoj mjeri određena korisna informacija pravilno opisuje određenu vrstu Web aplikacije. Podsjetnik na formulu na temelju koje se računa ukupna ocjena i donosi konačna odluka:

UKUPNA_OCJENA = k * op_ključne_riječi +u * op_uzorci_linkova + f * op_forme

pri čemu **k + u + f = 1** i pretpostavka da je **u > f > k**

Ispitivanjem se žele odrediti optimalne vrijednosti težišnih faktora **u, f** i **k**, gdje je **u** težišni faktor uzoraka linkova, **f** formi i **k** ključnih riječi. Pod optimalnim se podrazumijevaju vrijednosti za koje je greška u određivanju vrste Web aplikacije najmanja, odnosno ostavljaju najmanju nedoumicu pri donošenju konačne odluke.

Postupak ispitivanja vrste Web aplikacije se provodi u dva koraka:

1. Prikupljanje i vrednovanje *fingerprint* XML datoteka
2. Ispitivanje sustava i analiza rezultata

Prikupljanje i vrednovanje *fingerprint* XML datoteke je proces određivanja najbolje XML datoteke za određenu vrstu Web aplikacije, koja se koristi za daljnja ispitivanja. Ispitivanje sustava i analiza se provodi u više navrata (testova) za različite vrijednosti težišnih faktora kako bi se donio sud o najoptimalnijim težišnim faktorima.

Ispitivanje vrste Web aplikacije je usmjereno prema Web aplikacijama otvorenog koda. Zbog svoje popularnosti i zastupljenosti, odabrane su sljedeće dvije grupe i pripadne vrste Web aplikacija:

- Sustavi za upravljanje sadržajem - CMS (engl. *Content Manager System*)
  - Joomla!
  - Mambo
  - PHP-Nuke
  - Post Nuke
- Forumi
  - MyBB
  - PhpBB
  - UseBB
  - Bbpress
  - PunBB

Rezultati testiranja će se prezentirani u obliku tablica za svaku vrstu Web aplikacija, pri čemu će se analiza i komentari provoditi samo na reprezentativnim tablicama koje otkrivaju

određene posebnosti. Ostale tablice, odnosno rezultati ispitivanja se mogu pronaći u dodatku C.

5.1. Prikupljanje i vrednovanje *fingerprint* XML datoteka

Prije procesa ispitivanja, potrebno je imati kvalitetne *fingerprint* XML datoteke za svaku vrstu Web aplikacije. Kvalitetna XML datoteka treba sadržavati velik broj korisnih informacija kako bi se smanjila vjerojatnost donošenja krive ocjene pri ispitivanju određene vrste Web aplikacije.

Za prikupljanje XML datoteka se koristi generator *fingerprint* XML datoteka s dovoljno velikim pragom pametnog modula (više od 30). Nakon prikupljanja određenog broja XML datoteka iste vrste Web aplikacije s različitih adresa, slijedi vrednovanje XML datoteka. Cilj procesa vrednovanja je ocijeniti i izdvojiti najbolju XML datoteku za pojedinu vrstu Web aplikacije. Ocjenjivanje se temelji na formuli za izračun ukupne ocjene, pri čemu se prednost u potpunosti daje uzorcima linkova (**u**=1, **f**=0, **k**=0). XML datoteka s najboljom ocjenom se izdvaja kao predstavnik određene vrste Web aplikacije te se koristi u sljedećem koraku; ispitivanju sustava i analizi rezultata.

Sljedeći primjer opisuje postupak vrednovanja Joomla *fingeprint* XML datoteka. Na početku postupka, generator XML datoteka generira s istim pragom pametnog modula 4 uzorka: Joomla 1.a, Joomla 1.b, Joomla 1.c i Joomla 1.d. Podrijetlo prve tri datoteke je s različitih adresa na Internetu na kojima je prisutna Joomla Web aplikacija: Joomla 1.a s *http://www.lexnetcg.com* , Joomla1.b s *http://www.boks-savez.hr* i Joomla 1.c s *http://www.hru.hr*. Datoteka Joomla 1.d dobivena je prikupljanjem korisnih informacija s lokalno postavljene Joomla-e. Koristeći prikupljene datoteke, modul za otkrivanje vrste Web aplikacije vrši ocjenjivanje datoteka za 5 različitih adresa na kojima je prisutna Joomla. U tablici 5.1 su prikazane ocjene *fingerprint* datoteka za Joomla! Web aplikaciju.

Tablica 5.1. Ocjene različitih Joomla XML datoteka (u=1,f=0,k=1)

Adrese	**Ukupna ocijena**			
	Joomla 1.a	**Joomla1.b**	**Joomla 1.c**	**Joomla 1.d**
http://www.maneco-computers.com	84	100	53	7
http://myweb4all.com	100	100	60	10
http://www.jaic-consulting.com	100	100	57	7
http://amac.hrvati-amac.com	100	90	60	10
http://prijateljice.info/cms	100	90	63	9

Iz tablice je vidljivo kako XML datoteke Joomla 1.a i Joomla 1.b imaju najbolje ocjene. Ipak, zbog prosječno bolje ocjene, prednost u vrednovanju ima Joomla 1.a datoteka. Joomla 1.d ima uvjerljivo najgore ocjene za svih 5 adresa. Razlog tomu je podrijetlo datoteke. Naime, datoteka dobivena prikupljanjem informacija s nedavno instalirane Web aplikacije, u pravilu posjeduje manji broj uzoraka linkova i formi, što značajno utječe na ukupnu ocjenu. Tek instalirane Web aplikacije nemaju veliku količinu sadržaja, velik broj aktivnih modula i aktivnih formi, tako da se i ne može očekivati velika količina korisnih informacija.

Ako se pogleda interna struktura, Joomla1.a datoteka posjeduje oko 25 uzoraka linkova i desetak formi, dok Joomla 1.d posjeduje 9 uzoraka linkova i 5. formi. Kako količina korisnih informacija određuje kvalitetu XML datoteke, preporuka je prikupljanje korisnih informacija, odnosno generiranje XML datoteka s Web aplikacija koje su dostupne putem Interneta.

5.2. Ispitivanje sustava i analiza rezultata

Nakon prikupljanja kvalitetnih *fingerprint* XML datoteka za svaku vrstu Web aplikacije, provodi se ispitivanja sustava i analiza rezultata. Ispitivanje sustava se sastoji od tri različita testa za svaku vrstu Web aplikacije. Testovi se međusobno razlikuju po vrijednostima težišnih faktora, pri čemu se u obzir uzima pouzdanost određene vrste korisnih informacija. U razmatranju pouzdanosti pojedine vrste korisnih informacija, pretpostavlja se sljedeće:

- Uzorci linkova predstavljaju najpouzdaniju vrstu korisnih informacija Web aplikacija. Pretpostavka je da su uzorci linkova najmanje podložni promjenama od strane programera ili administratora Web aplikacija. Funkcionalnost većine Web aplikacije se temelji na linkovima, pa uvijek postoji prisutnost određenog broja uzoraka linkova
- Forme su pouzdana vrsta korisnih informacija, ali ne kao uzorci linkovi. Forme su podložnije promjena i po potrebi se mogu eliminirati od strane programera i administratora, ukoliko za njima nema potrebe.
- Ključne riječi nisu pouzdana vrsta korisnih informacija. Vrlo su podložne promjenama i najjednostavnije se mijenjaju, pa se može dogoditi da se ne ostavlja nikakav trag. Također, ključne riječi mogu upućivati na krivi trag, ako se ključna riječ spominje u nekom sasvim drugom kontekstu.

Uzimajući u obzir gornje pretpostavke o pouzdanosti korisnih informacija, definirana su tri različita testa prikazana tablicom 5.2.

Tablica 5.2. Testovi za ispitivanje sustava

test \ težišni faktor	K (ključne riječi)	F (forme)	U (uzorci linkova)
Test 1	0.1	0.2	0.7
Test 2	0	0	1
Test 3	0	0.4	0.6

Kao što se vidi, ključne riječi se uzimaju u obzir samo u prvom testu, gdje utječu na ukupnu ocjenu s minimalnih 10 %. Forme nemaju utjecaj u drugom testu, imaju manji utjecaj (20%) u prvom te značajan utjecaj u trećem testu (40%). Uzorci linkova su prisutni u svim testovima s najvećim utjecajem na ukupnu ocjenu, a posebno u drugom testu gdje se ukupna ocjena donosi samo na temelju uzoraka linkova (100%).

Sva tri testa se provode za svih devet navedenih vrsta Web aplikacija, pri čemu će se ispitivanje provesti na približno 6-10 adresa na kojima se pouzdano zna da je na njima prisutna određena vrsta Web aplikacije. Kako su rezultati za svaki test i za svaku vrstu Web aplikacije prikazani u tablicama, nepraktično je prikazivati i opisivati sve rezultate. Stoga, prikazati će se nekoliko reprezentativnih tablica s rezultatima na temelju kojih će se izvesti

određeni zaključci i opisati moguće iznimke. Ostali rezultati, odnosno rezultati testova za ostale vrste Web aplikacija su dostupni u dodatku C.

Prvo će se razmotriti rezultati iz grupe CMS Web aplikacija. CMS Web aplikacije ne posjeduju preveliku količinu sadržaja, pa postupak prikupljanje informacija ne traje predugo. Stoga, modul za prikupljanje informacija može koristiti visoki prag pametnog modula kako bi prikupio što veću količinu korisnih informacija. Velika količina korisnih informacija, omogućava modulu za otkrivanje vrste Web aplikacije precizniju procjenu, i u pravilu bolje rezultate.

Iz CMS grupe izdvojit će se Joomla. Ispitivanje sustava se provodi na deset adresa na kojima je prisutna Joomla. U tablici 5.3 su prikazani rezultati prvog testa dobiveni ispitivanjem Joomla Web aplikacije.

Tablica 5.3. Rezultati 1. testa za Joomla Web aplikaciju

Joomla! Test1	**Ukupna ocjena**								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://www.usdmo.hr	**100.0**	0.0	51.5	0.0	0.0	0.0	0.0	0.0	0.0
http://www.davidsengroup.com	**100.0**	0.0	90.0	0.0	0.0	0.0	0.0	0.0	0.0
http://www.mag.hr	**100.0**	0.0	40.0	0.0	0.0	0.0	0.0	0.0	0.0
http://www.hru.hr	**100.0**	0.0	14.0	0.0	0.0	0.0	0.0	0.0	0.0
http://www.bmbinvest.hr	**100.0**	0.0	58.2	0.0	0.0	0.0	0.0	0.0	0.0
http://www.boks-savez.hr	90.0	0.0	23.8	0.0	0.0	0.0	0.0	0.0	0.0
http://itsoft.hr	**100.0**	0.0	30.8	0.0	0.0	0.0	0.0	0.0	0.0
http://www.vk-osijek.hr	**100.0**	3.3	52.8	3.3	3.3	3.3	3.3	3.3	3.3
http://www.moslavina-info.com	**100.0**	0.0	70.0	0.0	0.0	0.0	0.0	0.0	0.0
http://www.ofm.hr	**100.0**	0.0	25.2	0	0	10.0	0.0	0.0	0.0

Rezultati prvog testa pokazuju sličnost između Joomla i Mambo Web aplikacije u vidu korisnih informacija. Ostale vrste Web aplikacije imaju neusporedivo manje ocjene ili ih uopće nemaju (0%), pa se s sigurnošću može reći da se ne radi o tim vrstama Web aplikacije. Iako se pouzdano zna je na adresama iz tablice prisutna Joomla, iznenađuje sličnost rezultata za Mambo i Joomla-u na adresama *http:/www.davidsengroup.com* i *http://www.moslavina-info.com*. Za prvu adresu Joomla ima tek nešto veći postotak – 100% naprema 90% od Mamba, dok je za drugu adresu razlika nešto veća – 100% naprema 70%. Zanimljiva je i adresa *http://www.boks-savez.org* koja ukazuje na neprisutnost ključnih riječi, zbog kojeg je ocjena 90%, umanjena za 10% koliko iznosi utjecaj ključnih riječi. Ako se pogledaju rezultati druga dva testa dobiva se uvid u kompletnu sliku. Tablica 5.4 prikazuje rezultate drugog testa , a tablica 5.5 trećeg testa.

Tablica 5.4. Rezultati 2. testa za Joomla Web aplikaciju

Joomla! Test2	**Ukupna ocjena**								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://www.usdmo.hr	**100.0**	0.0	45	0.0	0.0	0.0	0.0	0.0	0.0
http://www.davidsengroup.com	**100.0**	0.0	**100**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.mag.hr	**100.0**	0.0	50	0.0	0.0	0.0	0.0	0.0	0.0
http://www.hru.hr	**100.0**	0.0	20	0.0	0.0	0.0	0.0	0.0	0.0
http://www.bmbinvest.hr	**100.0**	0.0	76	0.0	0.0	0.0	0.0	0.0	0.0
http://www.boks-savez.hr	**100.0**	0.0	34	0.0	0.0	0.0	0.0	0.0	0.0
http://itsoft.hr	**100.0**	0.0	44	0.0	0.0	0.0	0.0	0.0	0.0
http://www.vk-osijek.hr	**100.0**	0.0	66	0.0	0.0	0.0	0.0	0.0	0.0
http://www.moslavina-info.com	**100.0**	0.0	**100**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.ofm.hr	**100.0**	0.0	36	0.0	0.0	0.0	0.0	0.0	0.0

Rezultati drugog testa otkrivaju da Joomla i Mambo imaju jako slične uzorke linkova, pogotovo ako se pogledaju adrese *http:/www.davidsengroup.com* i *http://www.moslavina-info.com*. Za te dvije adrese, Mambo i Joomla imaju ocjene podudarnosti uzoraka od 100%, pa se ne može donijeti konačna odluka o vrsti Web aplikacije samo na temelju uzoraka.

Tablica 5.5. Rezultati 3. testa za Joomla Web aplikaciju

Joomla! Test3	**Ukupna ocjena**								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://www.usdmo.hr	**100.0**	0.0	67.0	0.0	0.0	0.0	0.0	0.0	0.0
http://www.davidsengroup.com	**100.0**	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.mag.hr	**100.0**	0.0	30.0	0.0	0.0	0.0	0.0	0.0	0.0
http://www.hru.hr	**100.0**	0.0	12.0	0.0	0.0	0.0	0.0	0.0	0.0
http://www.bmbinvest.hr	**100.0**	0.0	55.6	0.0	0.0	0.0	0.0	0.0	0.0
http://www.boks-savez.hr	**100.0**	0.0	20.4	0.0	0.0	0.0	0.0	0.0	0.0
http://itsoft.hr	**100.0**	0.0	26.4	0.0	0.0	0.0	0.0	0.0	0.0
http://www.vk-osijek.hr	**100.0**	0.0	39.6	0.0	0.0	0.0	0.0	0.0	0.0
http://www.moslavina-info.com	**100.0**	0.0	60.0	0,0	0.0	0.0	0.0	0.0	0.0
http://www.ofm.hr	**100.0**	0.0	21.6	0.0	0.0	20.0	0.0	0.0	0.0

Ako se pogledaju rezultati trećeg testa, primjećuje se da osim uzoraka linkova, Mambo i Joomla imaju i slične forme. Ukupno gledajući, razlike u ocjenama su se povećale u korist Joomla-e u odnosu na drugi test. To nam govori da su Mambo forme i Joomla forme manje slične od njihovih uzoraka linkova. To potvrđuje i usporedba ocjena za *http://www.moslavina-info.com* za drugi i test gdje je ocjena za Mambo pala s 100% na 60%. Razlog je što na toj adresi ne postoji ni jedna forma koja se može naći u Mambo *fingerprint* XML datoteci. S druge strane, ako se pogleda *http:/www.davidsengroup.com* vidi se da se ocjena nije promijenila, što znači da je i za Joomla i Mambo pronađen isti broj podudarnih formi. Istraživanjem adrese *http:/www.davidsengroup.com* utvrđeno je da postoji samo jedna forma za prijavu korisnika u cijeloj aplikaciji. Usporedbom formi za prijava iz Joomla i Mambo fingerprint datoteke, primjećuje se velika sličnost između formi za prijavu. Kako se pri usporedbi formi koristi fleksibilno ocjenjivanje podudarnosti od 75%

polja, forme obiju Web aplikacija su ocijenjene podudarnima. Stoga, i u ovom slučaju se za adresu *http:/www.davidsengroup.com* ne može donijeti konačna odluka.

Analizom sva tri ispitivanja, zaključuju se da prvi test ostavlja najmanje nedoumica pri određivanju vrste Web aplikacije. Ključne riječi su pomogle u razrješavanju i zadnje nedoumice vezane uz adresu *http:/www.davidsengroup.com,*. Unatoč pretpostavci da su uzorci linkova i forme bitniji u konačnoj odluci, ponekad je korisno uzeti i ključne riječi u obzir.

Iz CMS grupe, također je zanimljiv treći test za Mambo čiji su rezultati prikazani u tablici 5.6.

Tablica 5.6. Rezultati 3. testa za Mambo Web aplikaciju

Mambo Test3	**Ukupna ocjena**								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://www.bilje.hr	45.0	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.ekopen.hr	62.6	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.kluberf.hr	39.6	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.universalmindband.com	34.2	0.0	60.0	0.0	0.0	0.0	0.0	0.0	0.0
http://www.hrvanje.com	77.2	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.tesla.hr	77.2	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.smz.hr	42.6	0.0	60.0	0.0	0.0	0.0	0.0	0.0	0.0
http://www.lilalo.com	70.0	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.opensourcecms.com	91.0	0.0	**100.0**	0.0	20.0	0.0	0.0	0.0	0.0
http://www.zapresic.hr	60.0	0.0	60.0	0.0	0.0	0.0	0.0	0.0	0.0

I u slučaju kad se ispitivanje vrši na adresama na kojima je prisutan Mambo, opet se primjećuje sličnost između Mambo-a i Joomla-a u vidu korisnih informaciju. Osim toga, zanimljivi rezultati za Mambo se dobiju na adresama *http://www.universalmindband.com* , *http://www.smz.hr* i *http://www.zapresic.hr* . Ocjena od 60% za Mambo ukazuje da nijedna forma na tim adresama nije podudarna se formama u Mambo *fingerprint* XML datoteci. Istraživanjem ove tri adrese otkrivena je samo jedna forma koja služi za prikupljanje glasova. Iako Mambo u svojoj fingeprint XML datoteci posjeduje formu za glasanje, broj glasačkih polja u formama za glasanje na gornjim adresama je znatno veći od broja glasačkih polja u formi u XML datoteci. Forma za glasanje iz XML datoteke je prikazana u slikom 5.1.

```
<form name="NoName" location="/">
     <from>/</from>
     <to>/index.php</to>
     <field name="task_button">Vote</field>
     <field name="task">vote</field>
     <field name="voteid">17</field>
     <field name="id">15</field>
</form>
```

Slika 5.1. Forma za glasanje u XML datoteci

Drugim riječima, forme prikupljene s adresa imaju veći broj polja s imenom *voteid* ,dok forma iz XML datoteke ima samo jedno polje *voteid*. Stoga će se usporedbom formi, forma proglasiti nepodudarnom što će u konačnici utjecati na ukupnu ocjenu u trećem testu.

Za ostale Web aplikacije iz CMS grupe; *PHP-Nuke* i *PostNuke* potrebno je istaknuti da imaju jedinstvene uzorke linkova. Kao posljedica, u drugom testu obje Web aplikacije za sve svoje adrese imaju maksimalne ocjene (100%), dok druge vrste Web aplikacija ne posjeduju ni jedan podudaran uzorak (0%). Par rezultata trećeg testa imaju ukupnu ocjenu 60%, što ukazuje na neprisutnost formi.

Sljedeća ispitna grupa Web aplikacija su forumi. Forumi, za razliku od CMS-ova, u pravilu posjeduju veću količinu podataka. Razlog leži u biti samog foruma; korisnici međusobno razmjenjuju informacije u različitim kategorijama i temama foruma. Ako forum ima velik broj korisnika, tada količina sadržaja, odnosno količina korisnih informacija može biti vrlo velika. Stoga je nužno da modul za prikupljanje informacija radi na manjem pragu pametnog modula kako bi u razumnom vremenskom periodu prikupio korisne informacije. Kao posljedica manjeg praga (oko 10), postoji veća mogućnosti pojave grešaka u određivanju vrste Web aplikacije.

Iz grupe foruma izdvojit će se UseBB. Ispitivanje sustava se provodi na devet adresa na kojima je prisutna UseBB Web aplikacija. U tablici 5.7 su prikazani rezultati prvog testa dobiveni ispitivanjem UseBB Web aplikacije.

Tablica 5.7. Rezultati 1. testa za UseBB Web aplikaciju

UseBB Test1	**Ukupna ocjena**								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://usebbzone.com	0.0	10.0	0.0	5.0	15.0	0.0	0.0	**100.0**	0.0
http://www.work-from-home-ic.com	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**100.0**	0.0
http://www.bamva.org/forum	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**100.0**	0.0
http://of.filestorages.com	0.0	0.0	0.0	0.0	0.0	0.0	0.0	30.0	0.0
http://www.windspots.com/forum	0.0	0.0	0.0	0.0	0.0	0.0	0.0	30.0	0.0
http://www.nationalcadstandard.org/forum	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**100.0**	0.0
http://www.thegreenhouseplace.com/usebb	0.0	0.0	0.0	6.6	10.0	0.0	6.6	**100.0**	0.0
http://redbox13.com/forum	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**100.0**	0.0
http://www.thegreenhouseplace.com/usebb	0.0	0.0	0.0	0.0	0.0	0.0	0.0	80.0	0.0

Iz rezultata prvog testa se primijeti da je za većinu adresa na kojima je prisutan UseBB forum, po sva tri kriterija, pravilno određena vrsta Web aplikacije (100%). Ostale vrste Web aplikacije ne prelaze postotak od 20%, pa se lako može zaključiti da se ne radi tim vrstama Web aplikacije. Ali također postoje iznimke na koje treba obratiti pozornost. Prva iznimka je adresa *http://www.thegreenhouse.com/usebb* koja ima nešto nižu ukupno ocjenu od 80%. Druge dvije iznimke su adrese *http://of.filestorages.com* i *http://windspots.com/forum* čijim su se ispitivanjem dobili relativno loši rezultati od 30% za UseBB. Kako bi se pronašao razlog nastanka iznimki, potrebno je razmotriti rezultate drugog testa prikazane u tablici 5.8 i rezultate treće testa prikazane u tablici 5.9.

Tablica 5.8. Rezultati 2. testa za UseBB Web aplikaciju

UseBB Test2	**Ukupna ocjena**								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://usebbzone.com	0	0	0	0	0	0	0	**100**	0
http://www.work-from-home-ic.com	0	0	0	0	0	0	0	**100**	0
http://www.bamva.org/forum	0	0	0	0	0	0	0	**100**	0
http://of.filestorages.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	**100**	0
http://www.thegreenhouseplace.com/usebb	0	0	0	0	0	0	0	**100**	0
http://redbox13.com/forum	0	0	0	0	0	0	0	**100**	0
http://www.thegreenhouseplace.com/usebb	0	0	0	0	0	0	0	**100**	0

Iz drugog testa je vidljivo da je uzrok loših rezultata za adrese *http://of.filestorages.com* i *http://windspots.com/forum* nepostojanje nijednog podudarnog uzorka linkova. Na prvi mah, to je malo zbunjujuće s obzirom na druge rezultate gdje je podudarnost uzoraka linkova 100%. Istraživanjem UseBB foruma s tih adresa, otkriveno je korištenje vlastitih imena parametara unutar linkova. Kao posljedica, uzorci linkova iz XML datoteke i oni prikupljeni sa adresa iznimki nisu podudarni. Standardnog UseBB uzorak linka je prikazan na slici 5.2:

```
<pattern>
      <url>/panel.php</url>
      <weight>12</weight>
      <params>
        <param name="act">register|login|staff|sendpwd|results</param>
        <param name="usebb_sid">ko2bf83ifc33thu48lq8rqhuo0</param>
      </params>
</pattern>
```

Slika 5.2. Standardni UseBB uzorak

Uzorak prikupljen sa adrese *http://of.filestorages.com* sadrži parametar *off_sid* umjesto standardnog *usebb_sid* parametra. Zbog načina usporedbe uzoraka linkova koji kaže da imena svih parametara moraju biti jednaka da bi uzorak bio podudaran, svi uzorci linkova iz XML datoteke i prikupljeni uzorci su nepodudarni. Stoga je logično da je za ove dvije adrese u drugom testu, gdje se gledaju samo uzorci linkova, ukupna ocjena 0%.

Tablica 5.9. Rezultati 3. testa za UseBB Web aplikaciju

UseBB Test3	**Ukupna ocjena**								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://usebbzone.com	0.0	20.0	0.0	10.0	30.0	0.0	0.0	**100.0**	0.0
http://www.work-from-home-ic.com	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**100.0**	0.0
http://www.bamva.org/forum	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**100.0**	0.0
http://of.filestorages.com	0.0	0.0	0.0	0.0	0.0	0.0	0.0	40.0	0.0
http://www.windspots.com/forum	0.0	0.0	0.0	0.0	0.0	0.0	0.0	40.0	0.0
http://www.nationalcadstandard.org/forum	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**100.0**	0.0
http://www.thegreenhouseplace.com/usebb	0.0	0.0	0.0	13.2	20.0	0.0	13.2	**100.0**	0.0
http://redbox13.com/forum	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**100.0**	0.0
http://www.thegreenhouseplace.com/usebb	0.0	0.0	0.0	0.0	0.0	0.0	0.0	60.0	0.0

Promatranjem trećeg testa, otkriva se da je razlog niže ocjene za adresu *http://www.thegreenhouse.com/usebb* nepronalazak nijedne podudarne forme. Mogući razlog nepronalaska nijedne forme je mali prag pametnog modula koja se koristi za prikupljanje korisnih informacija. Također valja zamijetiti relativno visoku ocjenu formi za PHP-Nuke za dvije adrese. To ukazuje na činjenicu, da su neke forme u PHP-Nuke *fingeprint* XML datoteci vrlo slične onima u UseBB XML *fingerprint* datoteci.

Ostali rezultati za vrste Web aplikacija iz grupe foruma su uglavnom monotoni. Tu se poglavito misli na drugi test gdje ne postoji niti jedna nedoumica, jer su ocjene 100% za jednu vrstu Web aplikacija, a za druge 0%. U trećem testu, postoje nešto viši postoci za vrste Web aplikacije koje nisu prisutne na testnim adresama (ne prelaze vrijednost od 20%). Ostali rezultati se mogu pogledati u dodatku C.

Analizom rezultata za svih devet ispitanih vrsta Web aplikacije, utvrđeno je da je prvi test za ispitivanje sustava dao najbolje rezultate. Rezultati svih testova su pokazali da je potrebno uzeti u obzir sve tri vrste korisnih informacija kako bi se dobili što bolji rezultati i razriješile nedoumice u slučaju iznimaka. Također je pokazano, da je potrebno poštivati omjer **u > f >k ,** iz razloga što najveću razliku između Web aplikacija čine uzorci linkova, pa forme i na kraju ključne riječi.

6.Zaključak

Mnoštvo današnjih sigurnosnih problema na Internetu je vezana uz područje sigurnosti Web aplikacija. Organizacije koje za svoje poslovne procese koriste Web aplikacije, ulažu sve više i više sredstava u različita rješenja kako bi zaštitili svoje Web aplikacije. Kao posljedica, dolazi do pojave novih i razvoja postojećih metoda i kontrola kako bi se doprinijelo podizanju razine sigurnosti Web aplikacija. U okviru ovog diplomskog rada je predložena i opisana metoda za otkrivanje poznatih ranjivosti u Web aplikacijama. Razlika ove metode sa poznatim metodama ispitivanja sigurnosti Web aplikacije leži u mogućnosti otkrivanja vrste Web aplikacije. Otkrivanje vrste Web aplikacije je postupak kojim se može znatno olakšati pronalazak poznatih ranjivosti za određenu vrstu Web aplikacije. Uspješnih otkrivanjem vrste Web aplikacije se može smanjiti trajanje postupka otkrivanja i povećati preciznost otkrivanja ranjivosti u određenoj Web aplikaciji. Stoga je naglasak cijelog rada bio na razvoju i ispitivanju sustava za otkrivanje vrste Web aplikacije.

Postupak otkrivanja vrste Web aplikacije se zasniva na usporedbi različitih vrsta korisnih informacija: uzoraka linkova, formi i ključnih riječi. Postupak također uzima u obzir pretpostavku da veća mjera pouzdanosti pojedine vrste korisnih informacija više utječe na odluku o vrsti Web aplikacije. Ispitivanjem sustava se nastojalo potvrditi istinitost pretpostavke i odrediti optimalne mjere važnosti faktora pojedine vrste korisnih informacija. Rezultati ispitivanja su potvrdili pretpostavku da najveću važnost imaju uzorci linkova, pa forme i na kraju ključne riječi. Iako rezultati ukazuju na činjenicu da se Web aplikacije najviše razliku po uzorcima linkova, postoje slučajevi gdje rezultati podudarnosti uzoraka linkova ostavljaju nedoumicu. Radi razrješavanja slučajeva u kojima se pojavljuje nedoumica, potrebno je uvrstiti i druge dvije vrste korisnih informacija u konačnu odluku, ali sa znatno manjim utjecajem. Analizom ispitnih rezultata, utvrđen je sljedeći omjer optimalnih vrijednosti:

- uzorci linkova utječu na konačnu odluku sa 70%,
- forme utječu na konačnu odluku sa 20% i
- ključne riječi utječu na konačnu odluka da 10%.

Sustav za otkrivanje vrste Web aplikacije, koristeći opisani postupak usporedbe, točno određuje vrstu za djelomičan skup vrsta Web aplikacija. Iako postoji manji broj iznimaka zbog kojih su rezultati lošiji, za većinu ispitanih adresa, sustav je pravilno odredio vrstu Web aplikacije s najvećom točnošću, odnosno ocjenom 100%.

Postoje i druge vrste Web aplikacija kojima nije moguće odrediti vrstu korištenjem opisanog postupka. Razlozi su sljedeći:

- Web aplikacije ne koriste standardne separatore parametara u linkovima. Takvi separatori se sastoje od niza znakova koje je teško razlikovati od sadržaja samog linka, pa stvaraju poteškoće u analizi linkova i organizaciji linkova u uzorke linkova.
- Web aplikacije uopće ne koriste linkove za prijenos vrijednosti. Ovo je čest slučaj kod Web aplikacija ostvarenih u ASP.Net okruženju, gdje se vrijednosti prenose unutar sjedničkih varijabli (engl. *session variable*) i to POST metodom.
- Web aplikacije koriste JavaScript za generiranje linkova. U ovom slučaju, linkovi se ne mogu prikupiti analizom HTML dokumenata.

Zbog navedenih nedostataka, sustav za otkrivanje vrste Web aplikacije je potrebno dodatno razvijati. Problem nestandardnih separatora se može riješiti vođenjem evidencije o separatorima unutar fingerprint XML datoteke. Na taj način se pri analizi linkova mogu korištenjem specifičnog separatora razdvojiti parametri i njihove vrijednosti, tj. linkovi se mogu organizirati u uzorke linkova kako bi se provela usporedba. Za Web aplikacije koje ne koriste linkove, potrebno je razviti drukčiji postupak otkrivanja vrste Web aplikacije. Postupak i dalje može koristiti ključne riječi i forme, ali se umjesto uzoraka linkova treba uvesti još jedna vrsta korisnih informacija. Primjerice, mogu se iskoristiti predefinirana imena sjedničkih varijabli i URL-i na koje se varijable proslijeđuju. Za Web aplikacije koje koriste JavaScript za generiranje linkova, potrebno je izraditi pomoćni modul koji će imati mogućnost dohvata informacija generiranih JavaScript-om. Postupak otkrivanja bi u ovom slučaju ostao isti.

7.Literatura

[1.] Wikipedia, *World Wide* Web, dostupno na Internet adresi: http://en.wikipedia.org/wiki/World_Wide_Web (5/7/2007)

[2.] Wikipedia, *Web application*, dostupno na Internet adresi: http://en.wikipedia.org/wiki/Web_application (5/7/2007)

[3.] CVE, *Common Vulnerabilities and Exposures CVE homepage,* dostupno na Internet adresi: http://cve.mitre.org/ (7/7/2007)

[4.] Web Application Security Consortium,*WASC homepage,* dostupno na Internet adresi: http://www.webappsec.org/ (10/2/2006)

[5.] Open Web Application Security Project, *OWASP homepage,* dostupno na Internet adresi: http://www.owasp.org (20/1/2007) :

[6.] Matija Zeman, *Sigurnost Web aplikacija zasnovanih na AJAX tehnologiji,* diplomski rad br. 1656., Fakultet Elektrotehnike i Računarstva, 2007.

[7.] Tony Northup: *MCAD/MCSD Self-Paced Training Kit: Implementing Security for Applications with Microsoft® Visual Basic® .NET and Microsoft Visual C#® .NET*,Microsoft Press, 2004.

[8.] Web Application Security Consortium (WASC), *Web Security Statistics,* dostupno na Internet adresi: http://www.webappsec.org/projects/statistics/ (6/7/2007)

[9.] Open Web application Security Project (OWASP), *OWASP Top Ten,* dostupno na Internet adresi: http://www.owasp.org/images/e/e8/OWASP_Top_10_2007.pdf (6/7/2007)

[10.] Steven M. Christey, *Vulnerability Type Distribution in CVE*, dostupno na Internet adresi: http://attrition.org/pipermail/vim/2006-September/001032.html (10/2/2007)

[11.] Web Application Security Consortium, *Web Application Security Consortium: Threat Clasification* (pdf), dostupno na Internet adresi: http://www.webappsec.org/projects/threat/v1/WASC-TC-v1_0.pdf (10/5/2007)

[12.] The Unofficial Cookie FAQ, dostupno na Internet adresi: http://www.cookiecentral.com/faq/ (14/2/2006)

[13.] Mitja Košek, *Session Fixation Vulnerability in Web-based Applications* (pdf), dostupno na Internet adresi: http://www.acrossecurity.com/papers/session_fixation.pdf (14/2/2006)

[14.] Kevin Spett, SPI Labs , *Cross-Site Scripting attacks* (pdf), dostupno na Internet adresi: http://www.spidynamics.com/whitepapers/SPIcross-sitescripting.pdf (16/2/2006)

[15.] Tim Newsham, *Format string attacks* (pdf), dostupno na Internet adresi: http://muse.linuxmafia.org/lost+found/format-string-attacks.pdf (16/2/2006)

[16.] Security Focus, *Trends in Web application security,* dostupno na Internet adresi: http://www.securityfocus.com/infocus/1809 (10/7/2007)

[17.] WhiteHat Security, *WhiteHat Security Web Application Security Risk Report* (pdf), dostupno na Internet adresi: http://www.whitehatsec.com/home/resources/whitepapers.html (10/7/2007)

[18.] NT objectives, *Application Vulnerability Scanners: Understanding Your Organization's Needs* (pdf), dostupno na Internet adresi: http://www.ntobjectives.com/download.php?filename=VW5kZXJzdGFuZGluZ09yZ2FuaXphdGlvbmFsTmVlZHMucGRm (15/7/2007)

[19.] NT objectives, *Web Application Exposure to Risk: Raising Awareness to Build Confidence and Improve Security* (pdf), dostupno na Internet adresi: http://www.ntobjectives.com/download.php?filename=V2ViQXBwbGljYXRpb25FeHBvc3VyZVdoaXRlUGFwZXIucGRm (15/7/2007)

[20.] NT objectives, *Web Application Security: Budgetary Considerations* (pdf), dostupno na Internet adresi: http://www.ntobjectives.com/download.php?filename=QnVkZ2V0YXJ5X0FzcGVjdHNfb2ZfV2ViX0FwcGxpY2F0aW9uX1NlY3VyaXR5LnBkZg== (15/7/2007)

[21.] WhiteHat Security, *Challenges of Automated Website Scanning* (pdf), dostupno na Internet adresi: http://www.whitehatsec.com/home/assets/presentations/challenges_of_scanning.pdf (16/7/2007)

[22.] Watchfire, *Web Application Security: Automated Scanning or Manual Penetration Testing* (pdf), dostupno na Internet adresi: http://www.watchfire.com/securityzone/whitepapers.aspx (20/7/2007)

[23.] Open Web application Security Project, *OWASP WebScarab Project,* dostupno na Internet adresi: http://www.owasp.org/index.php/ (20/1/2007)

[24.] OWASP WebScarab User documentation, dostupno na Internet adresi: http://dawes.za.net/rogan/webscarab/docs/ (21/1/2007)

[25.] "Marc Ruef", *The Attack Tool Kit Web Site ATK*, dostupno na Internet adresi: http://www.computec.ch/projekte/atk/ (27/1/2007)

[26.] Wapiti – Web application security auditor, dostupno na Internet adresi: http://wapiti.sourceforge.net/ (27/1/2007)

[27.] Dave Raget, *Overview of Tidy library,* dostupno na Internet adresi: http://www.w3.org/People/Raggett/tidy/ (20/3/2007)

Dodatak A: Engleski prijevodi korištenih termina

Gruba pretraga – Brute force

Administratorski zapis - Log

Umetanje nepostojećeg sadržaja - Content Spoofing

Izvršavanje napadačkog koda - Cross Site Scripting (XSS)

Izlistavanje mapad - Directory indexing

Razdvajanje HTTP odgovora - HTTP response splitting

Curenje informacija - Information Leakage

Izvršavanje naredbi OS-a - OS Commanding

Korištenje prečaca – Path traversal

SQL ubacivanje - SQL injection

Identifikator sjednice – Cookie

Poslovni procesi Web aplikacije – Web application workflow

Rukovatelj - Handler

Dodatak B: Primjer *fingerprint* XML datoteke

```
<?xml version=" 1.0" encoding="utf-8"?>
<fingerprint xmlns="WSAT_fingerprint_XML">
      <name>Drupal</name>
      <id>5</id>
      <keywords>
            <keyword>Drupal</keyword>
            <keyword>www.drupal.com</keyword>
      </keywords>
      <link_patterns>
            <separator>&</separator>
            <pattern>
                  <url>/</url>
                  <weight>28</weight>
                  <params>
                        <param name="destination">node|admin|
admin%2Fsettings                    </param>
                        <param name="q">node|user/register|
user/password|                      </param>
                  </params>
            </pattern>
            <pattern>
                  <url>/</url>
                  <weight>24</weight>
                  <params>
                        <param name="q">node|user/register|
user/password                       </param>
                  </params>
            </pattern>
            <pattern>
                  <url>/</url>
                  <weight>10</weight>
                  <params>
                        <param name="destination">node|admin|
admin%2Fsettings                    </param>
                  </params>
            </pattern>
      </link_patterns>
      <forms
            <form name="NoName" location="/">
                  <from>/</from>
                  <to>/?destination=node&q=node</to>
```

```
                <field name="op">Log in</field>
                <field name="name">on</field>
                <field name="form_id">user_login_block</field>
                <field name="pass">on</field>
            </form>
            <form name="NoName" location="/?q=user/">
                <from>/?q=user/register</from>
                <to>/?q=user/register</to>
                <field name="mail">on</field>
                <field name="op">Create new account</field>
                <field name="name">on</field>
                <field name="form_id">user_register</field>
            </form>
            <form name="NoName" location="/?q=user/">
                <from>/?q=user/password</from>
                <to>/?q=user/password</to>
                <field name="op">E-mail new password</field>
                <field name="name">on</field>
                <field name="form_id">user_pass</field>
            </form>
            <form name="NoName" location="/">
                <from>/?q=user</from>
                <to>/?q=user</to>
                <field name="op">Log in</field>
                <field name="name">on</field>
                <field name="form_id">user_login</field>
                <field name="pass">on</field>
            </form>
        </forms>
    </fingerprint>
```

Dodatak C: Ispitni rezultati

1. Ispitni rezultati za Mambo (CMS):

Mambo Test 1	**Ukupna ocjena**								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://www.bilje.hr	52.5	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.ekopen.hr	59.7	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.kluberf.hr	46.2	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.universalmindband.com	39.9	0.0	80.0	0.0	0.0	0.0	0.0	0.0	0.0
http://www.hrvanje.com	63.4	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.tesla.hr	63.4	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.smz.hr	49.7	0.0	80.0	0.0	0.0	0.0	0.0	0.0	0.0
http://www.lilalo.com	55.0	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.opensourcecms.com	79.5	0.0	**100.0**	0.0	10.0	0.0	0.0	0.0	0.0
http://www.zapresic.hr	70.0	0.0	80.0	0.0	0.0	00.0	0.0	0.0	0.0

Mambo Test 2	**Ukupna ocjena**								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://www.bilje.hr	75	0	**100**	0	0	0	0	0	0
http://www.ekopen.hr	71	0	**100**	0	0	0	0	0	0
http://www.kluberf.hr	66	0	**100**	0	0	0	0	0	0
http://www.universalmindband.com	57	0	**100**	0	0	0	0	0	0
http://www.hrvanje.com	62	0	**100**	0	0	0	0	0	0
http://www.tesla.hr	62	0	**100**	0	0	0	0	0	0
http://www.smz.hr	71	0	**100**	0	0	0	0	0	0
http://www.lilalo.com	50	0	**100**	0	0	0	0	0	0
http://www.opensourcecms.com	85	0	**100**	0	0	0	0	0	0
http://www.zapresic.hr	**100**	0	**100**	0	0	0	0	0	0

Mambo Test 3	**Ukupna ocjena**								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://www.bilje.hr	45.0	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.ekopen.hr	62.6	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.kluberf.hr	39.6	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.universalmindband.com	34.2	0.0	60.0	0.0	0.0	0.0	0.0	0.0	0.0
http://www.hrvanje.com	77.2	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.tesla.hr	77.2	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.smz.hr	42.6	0.0	60.0	0.0	0.0	0.0	0.0	0.0	0.0
http://www.lilalo.com	70.0	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0
http://www.opensourcecms.com	91.0	0.0	**100.0**	0.0	20.0	0.0	0.0	0.0	0.0
http://www.zapresic.hr	60.0	0.0	60.0	0.0	0.0	0.0	0.0	0.0	0.0

2. Ispitni rezultati za PHP – Nuke (CMS):

PHP-Nuke Test 1	**Ukupna ocjena**								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://www.zavidovici.info	0.0	0.0	0.0	0.0	80.0	0.0	0.0	0.0	0.0
http://www.lionskaptol.hr	1.8	5.4	0.0	1.8	**100.0**	0.0	1.8	3.6	0.0
http://www.gimnazija-djakovo.hr	1.6	5.0	0.0	1.6	**100.0**	3.2	1.6	3.2	0.0
http://www.alar-m.hr	5.0	8.6	5.0	6.8	**100.0**	8.6	6.8	8.6	5.0
http://www.meteori.org	6.6	10.0	5.0	6.6	**100.0**	10.0	6.6	8.2	5.0
http://www.croplanet.com	3.3	3.3	3.3	3.3	**100.0**	3.3	3.3	3.3	3.3
http://www.toposmedia.com	0.0	6.6	0.0	0.0	**100.0**	0.0	0.0	0.0	0.0
http://www.toposmedia.com	0.0	6.6	0.0	0.0	**100.0**	0.0	0.0	0.0	0.0
http://Wparanormal.com	0.0	5.0	0.0	0.0	**100.0**	0.0	0.0	0.0	0.0
http://www.astrostep.com	3.3	3.3	3.3	3.3	80.0	3.3	3.3	3.3	3.3

PHP-Nuke Test 2	**Ukupna ocjena**								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://www.zavidovici.info	0	0	0	0	**100**	0	0	0	0
http://www.lionskaptol.hr	0	0	0	0	**100**	0	0	0	0
http://www.gimnazija-djakovo.hr	0	0	0	0	**100**	0	0	0	0
http://www.alar-m.hr	0	0	0	0	**100**	0	0	0	0
http://www.meteori.org	0	0	0	0	**100**	0	0	0	0
http://www.croplanet.com	0	0	0	0	**100**	0	0	0	0
http://www.toposmedia.com	0	0	0	0	**100**	0	0	0	0
http://www.toposmedia.com	0	0	0	0	**100**	0	0	0	0
http://Wparanormal.com	0	0	0	0	**100**	0	0	0	0
http://www.astrostep.com	0	0	0	0	**100**	0	0	0	0

PHP-Nuke Test 3	**Ukupna ocjena**								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://www.zavidovici.info	0.0	0.0	0.0	0.0	60.0	0.0	0.0	0.0	0.0
http://www.lionskaptol.hr	3.6	10.8	0.0	3.6	**100.0**	0.0	3.6	7.2	0.0
http://www.gimnazija-djakovo.hr	3.2	10.0	0.0	3.2	**100.0**	6.4	3.2	6.4	0.0
http://www.alar-m.hr	0.0	7.2	0.0	3.6	**100.0**	7.2	3.6	7.2	0.0
http://www.meteori.org	3.2	10.0	0.0	3.2	**100.0**	10.0	3.2	6.4	0.0
http://www.croplanet.com	0.0	0.0	0.0	0.0	**100.0**	0.0	0.0	0.0	0.0
http://www.toposmedia.com	0.0	13.2	0.0	0.0	**100.0**	0.0	0.0	0.0	0.0
http://www.toposmedia.com	0.0	13.2	0.0	0.0	**100.0**	0.0	0.0	0.0	0.0
http://Wparanormal.com	0.0	10.0	0.0	0.0	**100.0**	0.0	0.0	0.0	0.0
http://www.astrostep.com	0.0	0.0	0.0	0.0	60.0	0.0	0.0	0.0	0.0

3.Ispitni rezultati za PostNuke (CMS):

PostNuke Test 1	Ukupna ocjena								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://www.ecs-online.biz	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0
http://earthedworld.co.uk	0.0	**100.0**	0.0	0.0	10.0	0.0	0.0	0.0	0.0
http://swanpool.org.uk	0.0	**100.0**	0.0	0.0	10.0	0.0	0.0	0.0	0.0
http://www.gastro-markt.hr	0.0	**100.0**	0.0	0.0	6.6	0.0	0.0	0.0	0.0
http://www.postnukepro.com	0.0	**100.0**	0.0	0.0	0.0	5.0	0.0	0.0	0.0
http://www.atomicwatermelon.com	0.0	80.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
http://www.deltasailing.com	0.0	**100.0**	0.0	0.0	10.0	0.0	0.0	0.0	0.0
http://www.philivision.com	0.0	90.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
http://falmouthbowly.org.uk	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0
http://wasteaction.org.uk	0.0	**100.0**	0.0	0.0	6.6	0.0	0.0	0.0	0.0

PostNuke Test 2	Ukupna ocjena								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://www.ecs-online.biz	0	**100**	0	0	0	0	0	0	0
http://earthedworld.co.uk	0	**100**	0	0	0	0	0	0	0
http://swanpool.org.uk	0	**100**	0	0	0	0	0	0	0
http://www.gastro-markt.hr	0	**100**	0	0	0	0	0	0	0
http://www.postnukepro.com	0	**100**	0	0	0	0	0	0	0
http://www.atomicwatermelon.com	0	**100**	0	0	0	0	0	0	0
http://www.deltasailing.com	0	**100**	0	0	0	0	0	0	0
http://www.philivision.com	0	**100**	0	0	0	0	0	0	0
http://falmouthbowly.org.uk	0	**100**	0	0	0	0	0	0	0
http://wasteaction.org.uk	0	**100**	0	0	0	0	0	0	0

PostNuke Test 3	Ukupna ocjena								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://www.ecs-online.biz	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0
http://earthedworld.co.uk	0.0	**100.0**	0.0	0.0	20.0	0.0	0.0	0.0	0.0
http://swanpool.org.uk	0.0	**100.0**	0.0	0.0	20.0	0.0	0.0	0.0	0.0
http://www.gastro-markt.hr	0.0	**100.0**	0.0	0.0	13.2	0.0	0.0	0.0	0.0
http://www.postnukepro.com	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0
http://www.atomicwatermelon.com	0.0	60.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
http://www.deltasailing.com	0.0	**100.0**	0.0	0.0	20.0	0.0	0.0	0.0	0.0
http://www.philivision.com	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0
http://falmouthbowly.org.uk	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0	0.0	0.0
http://wasteaction.org.uk	0.0	**100.0**	0.0	0.0	13.2	0.0	0.0	0.0	0.0

4. Ispitni rezultati za MyBB (Forum):

MyBB Test 1	**Ukupna ocjena**								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://homeundone.com	4.0	6.6	4.0	2.0	10.0	8.0	**100.0**	4.0	0.0
http://www.blakemiller.org/forum	0.0	0.0	0.0	0.0	0.0	0.0	**100.0**	0.0	0.0
http://www.letschatsports.com	0.0	0.0	0.0	0.0	0.0	0.0	**100.0**	0.0	0.0
http://www.letgamestalk.com	0.0	0.0	0.0	0.0	0.0	0.0	**100.0**	0.0	0.0
http://www.rc-forums.co.uk	0.0	0.0	0.0	0.0	0.0	0.0	**100.0**	0.0	0.0
http://www.debateforum.co.uk	0.0	0.0	0.0	0.0	0.0	0.0	**100.0**	0.0	2.1

MyBB Test 2	**Ukupna ocjena**								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://www.homeundone.com	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	0.0
http://www.blakemiller.org/forum	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	0.0
http://www.letschatsports.com	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	0.0
http://www.letgamestalk.com	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	0.0
http://www.rc-forums.co.uk	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	0.0
http://www.debateforum.co.uk	0.0	0.0	0.0	0.00	0.0	0.0	100.0	0.0	3.0

MyBB Test 3	**Ukupna ocjena**								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://www.homeundone.com	8.0	13.2	8.0	4.0	20.0	16.0	100.0	0.0	0.0
http://www.blakemiller.org/forum	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	0.0
http://www.letschatsports.com	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	0.0
http://www.letgamestalk.com	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	0.0
http://www.rc-forums.co.uk	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	0.0
http://www.debateforum.co.uk	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0	1.8

5. Ispitni rezultati za PhpBB (Forum):

PhpBB Test 1	Ukupna ocjena								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://public.carnet.hr/~joanicic/phpBB2	0.0	0.0	0.0	6.6	10.0	**100.0**	0.0	0.0	3.2
http://www.mojefinancije.net/phpBB2	0.0	0.0	0.0	10.0	5.0	**100.0**	0.0	0.0	0.0
http://www.sportskitrening.hr/phpBB2	0.0	0.0	0.0	6.6	10.0	**100.0**	0.0	0.0	3.2
http://www.slavonski-radio.com/phpbb2	0.0	0.0	0.0	6.6	10.0	**100.0**	0.0	0.0	3.2
http://www.solazur.ba/phpBB2	0.0	0.0	0.0	3.2	10.0	**100.0**	0.0	0.0	3.2
http://www.hrstud.hr/phpBB2	0.0	0.0	0.0	3.2	10.0	**100.0**	0.0	0.0	3.2
http://www.vcz.hr/phpBB2	0.0	0.0	0.0	6.6	6.6	**100.0**	0.0	0.0	3.2
http://hrvatska-kostajnica.hr/phpBB2	0.0	0.0	0.0	0.0	0.0	**100.0**	0.0	0.0	0.0
http://supertip.exxxtremebabes.com/phpBB2	0.0	0.0	0.0	0.0	0.0	**100.0**	0.0	0.0	0.0

PhpBB Test 2	Ukupna ocjena								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://public.carnet.hr/~joanicic/phpBB2	0	0	0	0	0	**100**	0	0	0
http://www.mojefinancije.net/phpBB2	0	0	0	0	0	**100**	0	0	0
http://www.sportskitrening.hr/phpBB2	0	0	0	0	0	**100**	0	0	0
http://www.slavonski-radio.com/phpbb2	0	0	0	0	0	**100**	0	0	0
http://www.solazur.ba/phpBB2	0	0	0	0	0	**100**	0	0	0
http://www.hrstud.hr/phpBB2	0	0	0	0	0	**100**	0	0	0
http://www.vcz.hr/phpBB2	0	0	0	0	0	**100**	0	0	0
http://hrvatska-kostajnica.hr/phpBB2	0	0	0	0	0	**100**	0	0	0
http://supertip.exxxtremebabes.com/phpBB2	0	0	0	0	0	**100**	0	0	0

PhpBB Test 3	Ukupna ocjena								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://public.carnet.hr/~joanicic/phpBB2	0.0	0.0	0.0	13.2	20.0	**100.0**	0.0	0.0	6.4
http://www.mojefinancije.net/phpBB2	0.0	0.0	0.0	20.0	10.0	**100.0**	0.0	0.0	0.0
http://www.sportskitrening.hr/phpBB2	0.0	0.0	0.0	13.2	20.0	**100.0**	0.0	0.0	6.4
http://www.slavonski-radio.com/phpbb2	0.0	0.0	0.0	13.2	20.0	**100.0**	0.0	0.0	6.4
http://www.solazur.ba/phpBB2	0.0	0.0	0.0	6.4	20.0	**100.0**	0.0	0.0	6.4
http://www.hrstud.hr/phpBB2	0.0	0.0	0.0	6.4	20.0	**100.0**	0.0	0.0	6.4
http://www.vcz.hr/phpBB2	0.0	0.0	0.0	13.2	13.2	**100.0**	0.0	0.0	6.4
http://hrvatska-kostajnica.hr/phpBB2	0.0	0.0	0.0	0.0	0.0	**100.0**	0.0	0.0	0.0
http://supertip.exxxtremebabes.com/phpBB2	0.0	0.0	0.0	0.0	0.0	**100.0**	0.0	0.0	0.0

6. Ispitni rezultati za BBpress (Forum):

Bbpress Test 1	**Ukupna ocjena**								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://flytheroad.com/blog/forums	0.0	0.0	0.0	**100.0**	0.0	0.0	0.0	0.0	14.0
http://www.speedieconsulting.co.uk/forum	0.0	0.0	0.0	80.0	0.0	0.0	0.0	0.0	14.0
http://onvertigo.com	0.0	0.0	0.0	**100.0**	0.0	10.0	0.0	0.0	0.0
http://forums.northbynorthwestern.com	0.0	0.0	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0
http://ronsmusings.com/bbpress	0.0	0.0	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0
http://www.txtpower.org/forums	0.0	0.0	0.0	**100.0**	0.0	0.0	0.0	0.0	14.0

Bbpress Test 2	**Ukupna ocjena**								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://flytheroad.com/blog/forums	0	0	0	**100**	0	0	0	0	20
http://www.speedieconsulting.co.uk/forum	0	0	0	**100**	0	0	0	0	20
http://onvertigo.com	0	0	0	**100**	0	0	0	0	0
http://forums.northbynorthwestern.com	0	0	0	**100**	0	0	0	0	0
http://ronsmusings.com/bbpress	0	0	0	**100**	0	0	0	0	0
http://www.txtpower.org/forums	0	0	0	**100**	0	0	0	0	20

BBpressTest 3	**Ukupna ocjena**								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://flytheroad.com/blog/forums	0.0	0.0	0.0	**100.0**	0.0	0.0	0.0	0.0	12.0
http://www.speedieconsulting.co.uk/forum	0.0	0.0	0.0	60.0	0.0	0.0	0.0	0.0	12.0
http://onvertigo.com	0.0	0.0	0.0	**100.0**	0.0	20.0	0.0	0.0	0.0
http://forums.northbynorthwestern.com	0.0	0.0	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0
http://ronsmusings.com/bbpress	0.0	0.0	0.0	**100.0**	0.0	0.0	0.0	0.0	0.0
http://www.txtpower.org/forums	0.0	0.0	0.0	**100.0**	0.0	0.0	0.0	0.0	12.0

7. Ispitni rezultati za PunBB (forum):

PunBB Test 1	Ukupna ocjena								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://www.umn-split.hr/forum	0.0	0.0	0.0	5.6	0.0	0.0	0.0	0.0	**100.0**
http://wufoo.com/forums	0.0	0.0	0.0	5.6	0.0	0.0	0.0	0.0	**100.0**
http://forum.download.ba	3.3	3.3	3.3	8.9	3.3	3.3	3.3	3.3	**100.0**
http://alexking.org/forums	0.0	0.0	0.0	6.3	0.0	0.0	6.3	0.0	**100.0**
http://www.libsna.org/forum	0.0	0.0	0.0	5.6	0.0	0.0	0.0	0.0	**100.0**
http://forum.strongspace.com	3.3	3.3	3.3	8.9	3.3	3.3	3.3	3.3	**100.0**
http://www.on-topics.com	3.3	3.3	3.3	8.9	3.3	3.3	3.3	3.3	**100.0**

PunBB Test 2	Ukupna ocjena								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://www.umn-split.hr/forum	0	0	0	8	0	0	0	0	**100**
http://wufoo.com/forums	0	0	0	8	0	0	0	0	**100**
http://forum.download.ba	0	0	0	8	0	0	0	0	**100**
http://alexking.org/forums	0	0	0	9	0	0	9	0	**100**
http://www.libsna.org/forum	0	0	0	8	0	0	0	0	**100**
http://forum.strongspace.com	0	0	0	8	0	0	0	0	**100**
http://www.on-topics.com	0	0	0	8	0	0	0	0	**100**

PunBB Test 3	Ukupna ocjena								
	Joomla	PostNuke	Mambo	bbPress	PHP-Nuke	PhpBB	MyBB	UseBB	PunBB
http://www.umn-split.hr/forum	0.0	0.0	0.0	4.8	0.0	0.0	0.0	0.0	**100.0**
http://wufoo.com/forums	0.0	0.0	0.0	4.8	0.0	0.0	0.0	0.0	**100.0**
http://forum.download.ba	0.0	0.0	0.0	4.8	0.0	0.0	0.0	0.0	**100.0**
http://alexking.org/forums	0.0	0.0	0.0	5.4	0.0	0.0	5.4	0.0	**100.0**
http://www.libsna.org/forum	0.0	0.0	0.0	4.8	0.0	0.0	0.0	0.0	**100.0**
http://forum.strongspace.com	0.0	0.0	0.0	4.8	0.0	0.0	0.0	0.0	**100.0**
http://www.on-topics.com	0.0	0.0	0.0	4.8	0.0	0.0	0.0	0.0	**100.0**