FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
SVEUČILIŠTE U ZAGREBU

DIPLOMSKI RAD br. 1712

AUTOMATIZIRANO TRAŽENJE SIGURNOSNIH PROPUSTA U WEB APLIKACIJAMA

Vanja Suhina

Zagreb, veljača 2008.

Zahvaljujem svima koji su mi pomogli u izradi ovog diplomskog rada, posebice doc. dr. sc. Zoranu Kalafatiću i mr. sc. Stjepanu Grošu na stručnom vodstvu i brojnim savjetima. Također veliko hvala roditeljima na beskrajnom strpljenu i potpori.

Sažetak

U ovom radu opisane su najkritičnije ranjivosti Web aplikacija te je pokazano kako alati za ispitivanje sigurnosti Web aplikacija vrše detekciju ranjivosti i koje su im mane. Zatim je predložen i opisan novi pristup detekciji ranjivosti temeljen na grupiranju stranica pomoću algoritama za raspoznavanje uzoraka. U praktičnom dijelu rada, takav pristup je i ostvaren te je prikazano kako se on uklopio u već postojeći sustav Web Security Assessment Tool (WSAT). Na kraju je opisano kako se sustav koristi te su prikazani eksperimentalni rezultati.

Abstract

This diploma thesis describes the most critical Web application vulnerabilities and it shows how the current Web application security assessment tools detect the vulnerabilities and where they fail. Then, the new approach to vulnerability detection based on algorithms for pattern recognition is proposed and described. We implemented the tool based on the mentioned idea and it is shown how it embeds in Web Security Assessment Tool (WSAT). In the end, it is described how the module is used and the experimental results are given.

Sadržaj

1. Uvod

Internet je postao opće prihvaćen način pristupu i razmijeni informacija. Zadnji statistički podaci pokazuju da se njime koristi više od 1,3 milijarde ljudi [1]. U najrazvijenijim zemljama se već sada Internetom koristi više od polovice populacije, između 50 i 70%, dok je u manje razvijenim zemljama primjetan velik rast korisnika u zadnjih desetak godina. Regije s najvećim rastom Internet korisnika između 2000. i 2007. godine su Bliski Istok s 920% rasta te Afrika s 880% rasta, a slijedi ih Središnja Amerika s gotovo 600%-tnim povećanjem korisnika [1]. Svi ti podaci pokazuju da je Internet globalno prihvaćen sustav za komunikaciju i pristup informacijama. Osim velikog rasta Internet korisnika, sama Internet mreža se također širi. Od početka devedesetih godina prošlog stoljeća kada se pojavio, Internet svake godine vrtoglavo raste i danas broji preko 2,67 milijardi Internet adresa [2].

Najraširenija usluga koja se koristi na Internetu je zasigurno World Wide Web (Web), a neke od jednako tako popularnih su elektronička pošta, dijeljenje datoteka (engl. file sharing) te instant komuniciranje (engl. chat). Web je sustav dokumenata koji mogu sadržavati tekst, slike i razni multimedijalni sadržaj, a navigacija između dokumenata se odvija putem poveznica (engl. hyperlinks). Pristup Web sadržaju se temelji na komunikaciji klijenata i poslužitelja. Klijenti pristupaju poslužiteljima koji su su dio Internetske računalne mreže i pomoću univerzalnog identifikatora resursa (engl. Uniform Resource Locator, URL) zahtijevaju određeni Web dokument. Poslužitelji odgovaraju klijentima sa sadržajem traženog Web dokumenta, najčešće u obliku HTML jezika, koji zatim Web preglednik na klijentu prikazuje korisniku u obliku Web stranice.

U početku su Web stranice bile statičke. Stranice su se uglavnom sastojale od teksta i slika i svaki put kada bi korisnik zatražio određenu stranicu, poslužitelj bi mu je vratio u identičnom obliku. Nije postojao način da se odredi kontekst u kojem je korisnik zatražio stranicu niti da se stranice prilagode različitim korisnicima. Upotrebom skriptnih jezika (poput JavaScript-a i VBScript-a) je porasla dinamičnost stranica. Postalo je moguće prilagoditi izgled stranice korisniku i prikazati elemente koji se mogu mijenjati s obzirom na korisnikovu okolinu. Korištenjem identifikatora sjednica pomoću tzv. kolačića (engl. cookies) Web stranice su postale sposobne odrediti kontekst u kojem je postavljen zahtjev te pratiti navigaciju korisnika i provoditi procese autentifikacije i autorizacije.

S vremenom se udio statičkih stranica drastično smanjio, jer su ih potisnule Web aplikacije. Web aplikacije su programi koji se izvršavaju na poslužiteljima i koji dinamički kreiraju Web stranice na svaki korisnikov zahtjev uzimajući u obzir kontekst i okolinu u kojoj je postavljen zahtjev. Tehnologije na klijentskoj strani pomažu pri komunikaciji s poslužiteljem i omogućuju prikaz bogatih multimedijalnih sadržaja, a aplikacije na poslužitelju prate korisnike i njihove zahtjeve te omogućuju kreiranje Web stranica prilagođenih korisniku. Korak dalje u dinamičnosti Weba nastaje sve raširenijom upotrebom AJAX tehnologije koja omogućuje da za promijene na Web stranici više nije potrebno dohvatiti cijelu novu stranicu, već se dohvaća samo dio koji je potrebno promijeniti. Proširila se upotreba novih bogatih Internet aplikacija: wiki aplikacije, blogovi, socijalne mreže. Zajednička svojstva svih tih aplikacija su bogato, interaktivno sučelje, mogućnost kreiranja sadržaja aplikacije i kontrole nad sadržajem, korištenje sadržaja drugih Web aplikacija unutar svoje aplikacije te stvaranje mreža korisnika aplikacije. Termin koji opisuje sva nova svojstva Web aplikacija je O'Reilly Media 2004. godine nazvao Web 2.0.

S pojavom dinamičkih Web stranica pojavili su se i prvi sigurnosni propusti vezani uz Web aplikacije. Do tada su sigurnosni propusti bili vezani samo uz Web poslužitelj ili uz operacijski sustav na kojem se Web poslužitelj izvršavao. Propusti nekad rezervirani isključivo za stolne aplikacije poput prelijeva spremnika (engl. buffer overflow) sada su se pojavili i na Webu. Sve veća kompleksnost aplikacija je dovela do većeg broja ranjivosti, a pojavile su se i ranjivosti vezane isključivo za Web. Danas sve velike tvrtke posjeduju svoje Internet stranice i omogućuju odvijanje dijela poslovanja preko Interneta. Napadi na takve aplikacije mogu dovesti do velikih novčanih gubitaka za tvrtke i njezine klijente, pada ugleda i gubljenja povjerenja svojih korisnika.

Ispitivanje sigurnosti Web aplikacija i procjena rizika postaje neizostavni dio svih projekata vezanih uz Internet. Ispitivanje mogu vršiti specijalizirani stručnjaci ili se ono može prepustiti jednom od alata za automatizirano traženje ranjivosti. Ručno ispitivanje je dugotrajno i skupo i teško može obuhvatiti sve slučajeve u kojima se pronalazi ranjivost. S druge strane, alati za automatsku detekciju ranjivosti nisu u stanju detektirati sve oblike ranjivosti i ponekad im je potrebna intervencija čovjeka. Pristup detekciji ranjivosti u ovom radu kombinira automatiziranost alata za detekciju i korisnikovo znanje kako bi pružilo bolje rezultate. Prvo su prikazane najkritičnije ranjivosti koje se pronalaze kod Web aplikacija te je opisano kako postojeći alati za analizu sigurnosnih propusta vrše detekciju ranjivosti. Zatim je predložen novi način vršenja detekcije ranjivosti za određene kategorije i opisani su problemi i prednosti takvog načina. Kao praktičan dio rada, napravljen je takav modul za već postojeći sustav Web Security Assessment Tool (WSAT) [3].

Struktura diplomskog rada je slijedeća. U drugom poglavlju su opisane najčešće kritične ranjivosti Web aplikacija. Prikazane su njihove podvrste i pokazano je kako funkcioniraju. Uz to je dan i primjer mehanizama za detekciju tih ranjivosti ukoliko ih je moguće detektirati alatima za automatsku detekciju. Kao primjeri su uzeti mehanizmi iz postojećih alata otvorenog kôda. Treće poglavlje sadrži prijedlog i opis novog pristupa detekciji ranjivosti i u njemu su objašnjeni popratni sadržaji vezani uz tematiku. U četvrtom poglavlju su opisane tehnologije koje se koriste u sustavu te su opisani korišteni postojeći moduli i eventualne izmjene u njima. Uz to je opisana i implementacija sustava za detekciju ranjivosti i njegovo korištenje. Peto poglavlje sadrži eksperimentalne rezultate. Rad završava sa zaključkom u šestom poglavlju i popisom literature.

2. Učestale ranjivosti Web aplikacija i njihova detekcija

Postoji puno organizacija koji se bavi sa sigurnošću računalnih sustava i Web aplikacija: SANS Institute [4], Web Application Security Consortium [5], Computer Emergency Response Team (CERT) [6], The Open Web Application Security Project (OWASP) [7], SecurityFocus [8], i dr. Svi oni posjeduju liste kategorija ranjivosti ili konkretnih ranjivosti pronađenih u sustavima. Centralizirana baza konkretnih ranjivosti pronađenih u sustavima je Common Vulnerabilities and Exposures (CVE) [9]. Na temelju te baze je napravljena studija [10] u kojoj su pronađene ranjivosti razvrstane u kategorije. Predstavnici OWASP zajednice su na temelju tog izvještaja i međusobne suradnje izradili dokument u kojem je nabrojano i objašnjeno deset najkritičnijih ranjivosti u Web aplikacijama [11]. Popis ranjivosti u ovom radu je preuzet iz tog dokumenta. Uz popis ranjivosti će u ovom poglavlju biti prikazano kako napadači mogu iskoristiti ranjivosti da bi proveli svoje zloćudne namjere te će biti pokazano kako postojeći alati za ispitivanje ranjivosti vrše detekciju. Popis se sastoji od slijedećih 10 tipova ranjivosti:

A1. Izvršavanje napadačkog kôda

A2. Propusti ubacivanja

A3. Izvođenje datoteka sa zlonamjernim sadržajem

A4. Nesigurna izravna referenca na objekt

A5. Krivotvorenje zahtjeva

A6. Ispuštanje informacija i neispravno rukovanje pogreškama

A7. Razbijena autentifikacija i kontrola sjednice

A8. Nesigurna kriptografska pohrana

A9. Nesigurna komunikacija

A10. Neuspješna zaštita pristupa URL-u

Neki od ovih tipova ranjivosti se mogu dodatno podijeliti u potkategorije pa će svaka potkategorija i iskorištavanje tog tipa ranjivosti biti posebno opisano. Uz pojedine tipove ranjivosti bit će dani i stvarni primjeri iskorištavanja ranjivosti.

Uz objašnjenje same ranjivosti bit će pokazano i kako neki poznati alati otvorenog kôda pristupaju problemu detekcije ranjivosti:

- Gamja
- Wapiti
- w3af
- Javascript XSS Scanner

Gamja [12] je alat za ispitivanje ranjivosti izvršavanja napadačkog kôda i SQL ubacivanja. Alat je trenutno u beta fazi izrade, moguće ga je pokretati iz komandne linije, a napisan je u programskom jeziku Perl.

Wapiti [13] je aplikacija za ispitivanje ranjivosti Web aplikacija i procjenu sigurnosti napisana u programskom jeziku Python. Wapiti je u mogućnosti naći pogreške kod rukovanja datotekama, ranjivosti izvršavanja napadačkog kôda i razne propuste ubacivanja (SQL, LDAP, CRLF,...).

W3af je kratica za alat pod punim imenom Web Application Attack and Audit Framework [14]. Jedan je od najdetaljnijih alata otvorenog kôda za procjenu sigurnosti Web aplikacija i pokriva vrlo širok spektar ranjivosti. Jedan je od rijetkih alata koji traži sve tipove ranjivosti izvršavanja napadačkog kôda i velik broj podtipova propusta ubacivanja. Moguće ga je podešavati kroz njegovu komandnu ljusku, a trenutno je u izradi i grafičko sučelje.

JavaScript XSS scanner je projekt od GNUCITIZEN-a [15] koji demonstrira da je moguće napisati alat za detekciju XSS ranjivosti u skriptnom jeziku JavaScript. Alat se zbog toga može izvršavati unutar svakog Web preglednika.

2.1. Izvršavanje napadačkog kôda

Izvršavanje napadačkog kôda (engl. Cross site scripting, XSS) je najčešća ranjivost kod Web aplikacija, a istraživanja su pokazala da je preko 80% svih Web aplikacija na Internetu ranjivo na XSS [16].

XSS je ranjivost u kojoj se maliciozan kôd ubacuje u HTML kôd prilikom dinamičkog kreiranja stranice. Maliciozan kôd se može ubacivati ili na poslužitelju gdje se dinamički kreira Web stranica koju je korisnik zatražio ili na klijentu prilikom kreiranja objektne reprezentacije dokumenta (engl. Document Object Model, DOM) i izvršavanja klijentskih skripti unutar korisnikovog Web preglednika.

U oba slučaja, problem je što se podaci koji se koriste prilikom kreiranja stranice, a mogu biti kontrolirani od strane napadača, ne provjeravaju niti filtriraju pa je napadač u mogućnosti umetnuti svoj kôd u klijentov Web preglednik koji će se izvršiti uz pomoć podržanog skriptnog jezika, najčešće JavaScript-a.

Postoje tri tipa ranjivosti izvršavanja napadačkog kôda, a razlikuju se po tome kako se maliciozan kôd ubacuje u stranicu i kada se prikazuje:

1. Privremeni ili Reflektirani XSS
2. Stalni ili Pohranjeni XSS
3. DOM ubacivanje ili Lokalni XSS

Svaki od tri tipa će biti posebno opisan i bit će objašnjeno kako se pojedini tipovi ranjivosti detektiraju.

Alati za detekciju ranjivosti (u daljnjem tekstu, skeneri) u procesu detekcije XSS ranjivosti ubacuju svoj kôd u ulazne parametre aplikacije i onda u odgovoru poslužitelja traže da li se pojavljuje ubačeni kôd te pokušavaju utvrditi da li će se kôd izvršiti. Najjednostavniji oblik XSS-a je reflektirani XSS jer se kôd odmah vraća u odgovoru na poslani zahtjev s kôdom. Problem kod pohranjenog XSS-a je da se kôd koji skener pokuša ubaciti u stranicu može pojaviti u nekom drugom odgovoru, na nekoj drugoj stranici pa skener mora voditi bilješke na kojoj stranici se pojavio kôd i s koje stranice je ubačen. Da bi skener bio u mogućnosti tražiti treći oblik XSS-a, DOM ubacivanje, mora imati mogućnost parsiranja JavaScript-a. Postoji dosta alata koji tvrde da detektiraju XSS ranjivosti, ali većina njih traga samo za najjednostavnijim oblikom, reflektiranim XSS-om.

2.1.1. Privremeni ili Reflektirani XSS

Ovo je najčešći i najjednostavniji primjer XSS ranjivosti. Napadač kreira poveznicu s malicioznim kôdom koji je usmjeren na ranjivu Web aplikaciju i vara korisnika da posjeti poveznicu. Poveznice se često objavljuju na popularnim forumima ili se šalju elektroničkom poštom. Kada korisnik posjeti poveznicu s malicioznim kôdom, ranjiva Web aplikacija uzima kôd iz vrijednosti parametra, ubacuje ga u HTML stranicu i vraća stranicu korisniku. Poveznica može izgledati kao u ispisu 2.1.

Ispis 2.1: Primjer malicioznog kôda u poveznici

```
http://www.somesite.com/search.php?query=<script>alert('xss')</script>
```

Na Web poslužitelju, u Web stranici *search.php*, neka ranjivi kôd izgleda kao u ispisu 2.2.

Ispis 2.2: Primjer ranjivog kôda na Web poslužitelju

```
<? echo("<p>Search results for query: ".$_GET['query'].".</p>"); ?>
```

U ovom primjeru kôda se vrijednost parametra *query* ubacuje u HTML kôd, bez da su se prethodno kôdirali posebni HTML znakovi. Rezultat takvog upita je kôd prikazan u ispisu 2.3.

Ispis 2.3: HTML kôd generiran prema ispisima 2.1 i 2.2

```
<p>Search results for query: <script>alert('xss')</script>.</p>
```

Element *script* koji je ubačen kroz *query* parametar označuje da se sadržaj elementa interpretira kao skripta te se ona izvršava unutar korisnikovog Web preglednika.

Kada napadač pokušava iskoristiti reflektiranu XSS ranjivost, mora natjerati korisnika da slijedi konstruiranu malicioznu poveznicu. Najčešći cilj takvih napada je krađa osobnih i povjerljivih podataka, krađa sjedničkog ključa i sl. Na primjer, ako napadač pronađe XSS ranjivost u nekoj bankovnoj aplikaciji koju koristi korisnik , može ga natjerati da slijedi poveznicu s malicioznim kôdom prikazanim u ispisu 2.4. Takav kôd služi za krađu sjedničkog ključa.

Ispis 2.4: Primjer kôda za krađu sjedničkog ključa

```
<script>var img=new Image();img.src='http://www.attacker.com/log.php?
   cookie=' +document.cookie</script>
```

Ako je korisnik autentificiran na bankovnu aplikaciju ili se njegovi podaci automatski šalju istoj kada joj se pristupa, podaci spremljeni u kolačiću će se poslati i pohraniti na napadačev Web poslužitelj. Napadač tada može preuzeti te podatke i s njima pristupiti bankovnoj aplikaciji kao da je on korisnik čiji je sjednički ključ saznao.

Reflektirani XSS je ujedno i najjednostavniji tip XSS ranjivosti za detekciju. Alati uglavnom prosljeđuju neku vrijednost parametrima aplikacije i onda u HTTP odgovoru provjeravaju pojavljivanje te vrijednosti.

XSS kôd koji Gamja ubacuje kroz zahtjeve je jednostavan i prikazan je u ispisu 2.5.

Ispis 2.5: Kôd koji Gamja koristi za detekciju XSS ranjivosti

```
">XSS_Check
```

Ako se taj XSS kôd pojavi u odgovoru, alat zaključuje da je aplikacija ranjiva na izvršavanje napadačkog kôda. Problem kod ovog alata je da metoda koristi samo nekoliko znakova koji mogu dovesti do XSS ubacivanja. Lako je moguće izvesti uspješne napade i bez znakova koji se koriste u ovom ispitivanju.

Javascript XSS Scanner je također sposoban detektirati samo ovaj oblik XSS-a. Za ispitivanje koristi dva jednostavna kôda za ubacivanje (ispis 2.6) i zatim traži pojavljivanje istog u odgovoru.

Ispis 2.6: Kôd koji koristi JavaScript XSS scanner za detekciju XSS ranjivosti.

```
/XSS_SCAN"><script>
/XSS_SCAN'><script>
```

Ovaj alat za razliku od Gamje ispituje da li je moguće ubaciti veći broj različitih znakova, ali je i dalje ne pokriva sve slučajeve u kojima dolazi do uspješnog napada. No, cilj ovog alata i nije sveobuhvatno testiranje već demonstracija da je takav alat moguće realizirati sa skriptnim jezikom JavaScript.

2.1.2. Stalni ili pohranjeni XSS

Cilj napada kod stalnog ili pohranjenog XSS-a je jednak kao i kod privremenog, tj. reflektiranog, a razlika je gdje i kako se maliciozan kôd vraća korisniku. Maliciozan kôd se ne mora odmah u HTTP odgovoru vratiti korisniku već se može spremiti u bazu podataka, na datotečni sustav ili na neku drugu lokaciju odakle se dohvaća prilikom kreiranja stranice. Moguće je da se maliciozan kôd odmah prilikom ubacivanja vrati korisniku, ali snaga ovog napada je da se taj isti kôd vraća korisnicima svaki put kada oni zatraže određenu stranicu unutar koje se kôd ugrađuje. Više nije potrebno natjerati korisnika da posjeti poveznicu s malicioznim kôdom, dovoljno je da korisnik posjeti Web stranicu ranjive aplikacije u koju je napadač prethodno ubacio maliciozan kôd. Ovaj tip napada cilja širu populaciju korisnika i zbog toga je opasniji. Napadaču je dovoljno da jednom ubaci maliciozan kôd kroz ranjivu Web stranicu, i kôd će se zatim slati svakom posjetitelju stranice. Primjer ubacivanja kôda kao komentara na Web stranici je prikazan u ispisu 2.7. Maliciozan kôd se može vratiti korisniku kroz neku javnu stranicu dostupnu svima ili može ciljati nekog određenog korisnika ili administratora aplikacije.

Ispis 2.7: Primjer ubacivanja napadačkog kôda

```
http://www.someblog.com/post.php?comment=<script>alert('xss')</script>
```

Maliciozan kôd koji se koristi prilikom ovog napada može biti sličan kôdovima kod reflektiranog XSS-a. Pohranjivanje kôda na poslužitelju i vraćanje kôda svim korisnicima koji pristupe određenom resursu čini ovaj tip napada vrlo raširenim. Napadač može iskoristiti ovaj tip ranjivosti za prikupljanje korisnikovih privatnih podataka, krađu sjedničkih ključeva, modificiranje izgleda stranice i provođenje *phishing* napada. Postoji nekoliko XSS crva koji koriste ovaj tip XSS ranjivosti za infekciju korisnika i propagiranje kroz mrežu.

Detekcija pohranjenog XSS-a je složeniji zadatak od detekcije reflektiranog XSS-a. Alati u ovom slučaju moraju biti u stanju detektirati ubačeni kôd na bilo kojoj stranici na koju naiđu te moraju voditi evidenciju s koje stranice i kroz koji parametar je taj kôd ubačen u aplikaciju.

Wapiti je alat koji je u stanju detektirati reflektirani i pohranjeni XSS. Prije ubacivanja kôda, kôd se prilikom svakog zahtjeva posebno generira kako bi se, kada se pronađe u nekom odgovoru, moglo prepoznati kroz koju stranicu i parametar je ubačen. Wapiti koristi kôd čiji je oblik prikazan u ispisu 2.8.

Ispis 2.8: Oblik kôda koji Wapiti koristi za detekciju XSS-a

```
<script>var wapiti_[0-9a-h]+_[0-9a-h]+=new Boolean\(\);</script>
```

Prva grupa znakova iza dijela *wapiti_* je naziv stranice s koje je došao zahtjev, prikazan u heksadecimalnom obliku. Druga grupa znakova je ime parametra kroz koji se kôd ubacio također prikazan u heksadecimalnom obliku. Kada alat u nekom odgovoru pronađe dio kôda koji odgovara kôdu u ispisu 2.8, alat zaključuje da postoji XSS ranjivost, a iz samog pronađenog kôda saznaje s koje je stranice i kroz koji parametar kôd ubačen u aplikaciju.

Na sličan način i w3af generira kôd koji se ubacuje u ulazne parametre Web aplikacije. Pri kreiranju XSS kôda za ubacivanje koristi razne oblike kôda pritom koristeći jednostruke navodnike, dvostruke navodnike ili oblik bez navodnika (ispis 2.9).

Ispis 2.9: Oblik kôda koji w3af koristi za detekciju XSS-a

```
<SCRIPT>alert2('RANDOMIZE')</SCRIPT>
javascript:alert('RANDOMIZE');
JaVaScRiPt:alert('RANDOMIZE');
javas\tcript:alert('RANDOMIZE');
<SCRIPT>a=/XSS/\nalert(a.source)</SCRIPT>RANDOMIZE
javascript:alert("RANDOMIZE");
JaVaScRiPt:alert("RANDOMIZE");
<SCRIPT>alert("RANDOMIZE")</SCRIPT>
javas\tcript:alert("RANDOMIZE");
```

Prije samog ubacivanja kôda, termin *RANDOMIZE* se mijenja sa slučajnim brojem, kako bi se kasnije moglo odrediti s koje stranice i kroz koji parametar je kôd ubačen.

W3af je kompleksniji alat od prethodno spomenutih i koristi naprednije metode od običnog traženja kôda u odgovorima. Odgovori se analiziraju i pokušava se odrediti koji su filtri doprinijeli sprječavanju XSS ranjivosti. Te se informacije onda mogu iskoristiti za kreiranje novih kôdova za ubacivanje i za zaobilaženje mehanizma filtriranja u Web aplikaciji.

2.1.3. DOM Ubacivanje ili Lokalni XSS

Za razliku od prva dva tipa, ovaj tip XSS ranjivosti se ne oslanja na ubacivanje malicioznog kôda u HTML kôd na poslužitelju. Kôd se ubacuje na klijentu, kroz Web preglednik koji je odgovoran za izvršavanje JavaScript funkcija. Pretpostavimo da postoji skripta u HTML kôdu koji izgleda kao u ispisu 2.10.

Ispis 2.10: Primjer HTML kôda ranjivog na DOM ubacivanje

```
<p>You came from: <script>document.write(document.referrer)</script>/p>
```

Funkcija skripte je da ispiše s koje je stranice korisnik posjetio trenutnu stranicu. *Document.referrer* je Document Object Model (DOM) komponenta koja se popunjava podacima prilikom kreiranja Web stranice u Web pregledniku. Zapravo se prolazi kroz HTTP zaglavlja i traži zaglavlje *Referer* te se ta vrijednost kopira u DOM. Ako napadač uspije prisiliti korisnika da posjeti ovakvu stranice s neke stranice koja ima maliciozan kôd u

URL-u (kao u ispisu 2.11.), maliciozan kôd će se prvo spremiti u HTTP zaglavlje *Referer*, a zatim će se u korisnikovom Web pregledniku pomoću JavaScript-a ubaciti u HTML kôd.

Ispis 2.11: Primjer stranice s malicioznim kôdom u URL-u

```
http://www.website.com/somepage.html?<script>alert('xss')</script>
```

Korisnik koji posjeti ovakvu stranicu će kao krajnji rezultat dobiti HTML kôd čiji je dio prikazan u ispisu 2.12.

Ispis 2.12: dio HTML kôda nakon DOM ubacivanja

```
<p>You came from:
http://www.website.com/somepage.html?<script>alert('xss')</script></p>
```

Rizične su sve JavaScript funkcije koje pristupaju dijelovima URL-a i HTTP zaglavljima i ubacuju te vrijednosti u HTML kôd. *Document.referrer* nije jedini DOM objekt koji se može iskoristiti za DOM ubacivanje. Neki od pogodnih su: *document.URL, document.URLUNencoded, document.location, window.location.*

Prilikom napada pomoću DOM ubacivanja, napadač može nanijeti najviše štete ako pronađe ranjivost u poznatoj datoteci na datotečnom sustavu. U tom slučaju napadač može prisiliti korisnika da otvori ranjivu stranicu s datotečnog sustava i proslijediti joj maliciozan kôd. Kada se lokalna datoteka otvori u Web pregledniku, ona se otvara s višim pravima nego što su prava datoteke otvorene s Interneta. Kôd koji se izvršava kroz nju onda također ima veća prava i može nanijeti veću štetu.

Detekcija DOM ubacivanja se provodi na drugačiji način od prethodno prikazanih. W3af u potrazi za DOM ubacivanjem prolazi kroz sve odgovore i traži skripte koje koriste neku od potencijalno ranjivih funkcija prikazanih u ispisu 2.13 s jednom od potencijalno ranjivih varijabli prikazanih u ispisu 2.14.

Ispis 2.13: Popis potencijalno ranjivih JavaScript funkcija na DOM ubacivanje

```
document.write
document.writeln
document.execCommand
document.open
window.open
eval
window.execScript
```

Ispis 2.14: Popis potencijalno ranjivih varijabli na DOM ubacivanje

```
document.URL
document.URLUnencoded
document.location
document.referrer
window.location
```

Ako se pronađe skripta koja kombinira potencijalno ranjive funkcije i varijable, alat zaključuje da tu može doći do DOM ubacivanja.

2.1.4. Primjeri iskorištavanja XSS ranjivosti – XSS crvi

U ovom poglavlju je dan prikaz nekoliko XSS crva koji iskorištavaju XSS ranjivosti. Pokazano je da je moguće napisati crve u programskom jeziku JavaScript, koji se propagiraju kroz mrežu koristeći XSS ranjivosti.

Prvi velik i najpoznatiji crv ovog tipa je Samy crv [17] koji se proširio po portalu MySpace.com u listopadu 2005. godine. Koristio je nesavršenosti u mehanizmu za filtriranje koji je trebao spriječiti bilo kakav unos malicioznog HTML kôda. Autor crva Samy Kamkar je uspio zaobići pravila filtriranja i postaviti prvi primjerak malicioznog kôda na Web stranicu sa svojim korisničkim profilom. Svaki korisnik koji je pristupio autorovom profilu bi u odgovoru dobio maliciozan kôd pa je to primjer stalnog ili pohranjenog XSS napada. Maliciozan kôd je bio napisan tako da sam sebe kopira na profil svakog korisnika koji bi posjetio zaraženi profil pa se ubrzo počeo širiti među korisnicima. U periodu od 24 sata, prije nego što je portal morao biti ugašen, crv se proširio na preko 1.000.000 korisničkih profila. Time je Samy crv postavio rekord u brzini širenja, ostavljajući daleko za sobom neke klasične crve kao što su Blaster, Slammer ili Code Red. Dobra strana u priči je što maliciozan kôd koji se širio i nije bio maliciozan. Jedino što je kôd radio je dodavanje Samyja u korisnikovu listu prijatelja i dodavanje poruke na korisnički profil koja je glasila: "but most of all, Samy is my hero."

Drugi konceptualni primjer XSS crva je napisao talijanski stručnjak za sigurnost Rosario Valotta koji je napisao prvi XSS crv koji se širi preko četiri različite domene, popularnih talijanskih servisa za elektroničku poštu [18]. Nduja Connection crv cilja na Libero.it, Tiscali.it, Lycos.it i Excite.com i iskorištava XSS ranjivosti pronađene u svakom od navedenih Web aplikacija. XSS ranjivosti dozvoljavaju ubacivanje malicioznog kôda u tijelo elektroničke pošte pa se kôd izvršava ako korisnik samo otvori elektroničku poruku putem Web preglednika. Crv zatim prikuplja sve elektroničke poruke, iz njih izvlači adrese novih korisnika i dalje se propagira. Kako je ovo samo koncept, crv ne poduzima nikakve dodatne maliciozne akcije.

U srpnju 2007. godine, Benjamin Flesch je napisao prvi prijateljski nastrojen XSS crv koji se širi po mreži Wordpress blogova i pokušava instalirati sigurnosne zakrpe za neke ranjivosti [19]. Crv za propagiranje koristi AJAX i iskorištava XSS i CSRF ranjivosti.

Sve češće smo svjedoci pojavljivanja XSS crva u javnosti i pitanje je vremena kada će se pojaviti jedan sa stvarno malicioznim kôdom i napraviti veliku štetu.

2.2. Propusti ubacivanja

Propusti ubacivanja (engl. injection flaws) su druga najčešća ranjivost kod Web aplikacija. Samo SQL ubacivanje zauzima 13% svih ranjivosti prijavljenih u 2006. [16] i smatra se da je svaki četvrta Web aplikacija ranjiva na SQL ubacivanje. Drugi propusti ubacivanja, kao što su LDAP, XPath i CRLF ubacivanja samo povećavaju tu brojku i čine propuste ubacivanja jako raširenim i pogodnim za iskorištavanje.

Propusti ubacivanja se događaju kada se korisnikovi podaci koriste za kreiranje upita i naredbi bez provjere da li sadrže posebne znakove koji bi mogli biti korišteni u maliciozne svrhe. Maliciozan kôd može promijeniti sadržaj i smisao upita, ili naredbe, i poslužitelj onda može izvršavati korisnikove naredbe. Najčešći tip ubacivanja je SQL ubacivanje, ali postoje

i razni drugi tipovi: LDAP, XPath, XML, XSLT, OS. U ovom poglavlju, bit će opisani slijedeći tipovi propusta ubacivanja:

1. SQL ubacivanje
2. XPath ubacivanje
3. LDAP ubacivanje
4. CRLF ubacivanje
5. Ubacivanje OS naredbi
6. SSI ubacivanje
7. MX ubacivanje

Postoje i drugi tipovi propusta ubacivanja, koji su slični većini ovdje navedenih, ali nisu ovdje obrađeni.

Detekcija propusta ubacivanja je kod gotovo svih alata za detekciju ranjivosti (skenera) jednaka. Skener šalje kroz ulazni parametar kôd koji bi trebao uzrokovati grešku na poslužitelju i onda se u odgovoru poslužitelja traži pojavljivanje neki od učestalih tekstova s opisom greške. U ovom poglavlju će svi tipovi propusta ubacivanja biti opisani posebno i biti će prikazano koje su to najčešći dijelovi teksta koji indiciraju da je došlo do greške na poslužitelju.

Većina alata za detekciju propusta ubacivanja traži samo propuste kod SQL ubacivanja jer je to najčešći tip ove ranjivosti, neki alati još traže ranjivosti LDAP i XPath ubacivanja, a rijetki su oni koji su u stanju detektirati ostale tipove.

Unatoč sličnostima detekcije pojedinih tipova propusta ubacivanja, za svaki će tip biti posebno pokazane greške koje se koriste pri detekciji.

2.2.1. SQL ubacivanje

SQL ubacivanje (engl. SQL injection) se događa kada se korisnički podaci koriste za formiranje SQL upita bez da su prethodno filtrirani i provjereni po tipu. To može dovesti do mijenjanja značenja SQL upita. Pogledajmo ispis 2.15. koji prikazuje primjer kreiranja SQL upita s korisničkim podacima.

Ispis 2.15: Primjer kreiranja SQL upita

```
query="SELECT data FROM users WHERE name='" +$name + "' AND pass='" +
$pass +"';"
```

Ako se parametri *name* i *pass* ne provjeravaju za specijalne znakove, napadač može promijeniti značenje upita koristeći tekst prikazan u ispisu 2.16.

Ispis 2.16: Primjer teksta koji mijenja značenje SQL upita iz ispisa 2.15.

```
a' OR '1'='1
```

Ovaj ulazni podatak mijenja smisao SQL upita tako da upit uvijek vraća pozitivnu vrijednost iako parametar *pass* nije točan pa se s time potpuno zaobilazi proces autentifikacije korisnika. Ako napadač ne posjeduje ime niti jednog korisnika, može iskoristiti isti takav pristup da se autentificira kao prvi korisnik iz baze podataka.

Detekcija navodnika koji mogu promijeniti smisao upita sama po sebi nije dovoljna jer postoje slučajevi kada se SQL upit može promijeniti koristeći samo dozvoljene znakove. U ispisu 2.17 je prikazan primjer kreiranja novog SQL upita, a u ispisu 2.18 je prikazano kako napadač može promijeniti upit.

Ispis 2.17: Primjer kreiranja SQL upita

```
query = "SELECT data FROM users WHERE userid=" + $userid
```

Ispis 2.18: Primjer teksta koji mijenja značenje SQL upita iz ispisa 2.17

```
123 or userid is not null
```

Ne provjeravanje tipa ulaznog podataka za parametar *userid* se u ovom slučaju smatra ranjivošću.

Napadi na propuste ubacivanja imaju drugačije ciljeve nego napadi na XSS ranjivosti. Namjera kod mijenjanja ulaznih podataka je promjena značenja upita i pristup privatnim informacijama koje su spremljene u bazi podataka. Cilj je krađa povjerljivih informacija, korisnikovih lozinki ili zaobilaženje sustava autentifikacije.

Prilikom mijenjanja ulaznih podataka, napadač će često promijeniti SQL upit tako da on više nije valjan. U slučaju da korištenje stranicama s greškama nije riješeno kako bi trebalo biti, poruke SQL poslužitelja sa ili bez dijelova SQL upita se mogu vratiti napadaču i prikazati na ekranu. Napadač nadalje može koristiti te informacije da kreira bolje ulazne podatke koji neće narušiti valjanost upita i koji će vratiti povjerljive podatke.

Ako je pravilno podešeno korištenje stranica s greškama, nevaljani upiti neće otkriti nikakve dodatne informacije o postojećem sustavu i tipu greške, ali napadač može primijeniti drugu tehniku za otkrivanje podataka. U slučaju da se SQL ubacivanje i dalje događa, ali se poruke o greškama ne prikazuju korisniku, napadač kreira dodatan uvjet na SQL upit koji može biti istinit ili ne. U slučaju da je uvjet istinit, prikazat će se normalna stranica, a u slučaju da nije istinit, prikazat će se npr. stranica s općenitom greškom. Napadač je sada u stanju postavljati pitanja nad bazom i detektirati kada je odgovor istinit ili ne. Ova tehnika napada se zove slijepo SQL ubacivanje [20], a pomoću njega je moguće čak izlistati cijeli sadržaj baze podataka.

U slučajevima kada se kao SQL poslužitelj koristi Microsoft SQL poslužitelj (MS SQL), moguće je da SQL ubacivanje može dovesti do izvršavanja naredbi operacijskog sustava na kojem se izvršava Web poslužitelj. U MS SQL poslužitelju postoji sistemska procedura *xp_cmdshell* [21] koja se može koristiti ako korisnik ima dovoljna prava u bazi podataka. Procedura se koristi za proslijeđivanje komandi operacijskom sustavu Windows. Napad se može provesti tako da se napadačeve komande proslijeđuju operacijskom sustavu, a da se njegov odgovor vraća natrag napadaču. Naizgled djeluje kao da je napadač spojen direktno na Web poslužitelj i izvršava naredbe po želji.

Detekcija SQL ubacivanja se kod većine alata vrši na sličan način. Alati pokušavaju uzrokovati greške na poslužitelju i u odgovorima se traže učestale greške SQL poslužitelja.

Gamja traži sve propuste ubacivanja na jednak način. Kao kôd za ubacivanje koristi jednostruki navodnik i par slova i zatim traži jednu od slijedećih riječi u odgovoru (ispis 2.19).

Ispis 2.19: Riječi koje Gamja traži kod detekcija propusta ubacivanja

```
Warning:
JDBC
SQL syntax
ODBC
'SQL
error
```

Ovakvi izrazi o greškama su prekratki i mogu biti pronađeni na stranicama koje nisu uzrok greške na poslužitelju.

Wapiti također koristi jednak kôd za ispitivanje svih tipova propusta ubacivanja (ispis 2.20).

Ispis 2.20: Kôd koji koristi Wapiti pri pokušaju generiranja greške na poslužitelju

```
\xbf'"(
```

Prilikom analize odgovora, traže se neke uobičajene poruke o greškama i na temelju njih odlučuje da li je došlo do SQL, XPath ili LDAP ubacivanja. Ako je pronađena poruka o greški karakteristična za SQL ubacivanje, wapiti odlučuje i koji poslužitelj se koristi s Web aplikacijom (MySQL ili MS SQL). Popis poruka za SQL ubacivanje prikazan je u ispisu 2.21.

Ispis 2.21: Tipične poruke o greškama koje Wapiti koristi za detekciju propusta SQL ubacivanja

```
You have an error in your SQL syntax
supplied argument is not a valid MySQL
[Microsoft][ODBC Microsoft Access Driver]
java.sql.SQLException: Syntax error or access violation
```

W3af koristi jednak pristup kao i dosad spomenuti alati, ali posjeduje veću bazu znanja o greškama i postojećim poslužiteljima. Kao kôd za ubacivanje koristi jednostavan fragment (ispis 2.22), a za svaku poruku o greški veže pripadni poslužitelj baze podataka koji obično generira takvu poruku (ispis 2.23).

Ispis 2.22: Kôd koji w3af koristi pri pokušaju generiranja greške na poslužitelju

```
d'z'0
```

Ispis 2.23: Popis poruka o greškama i pripadajućih poslužitelja baza podataka

```
[IBM][CLI Driver][DB2                    : IBM db2 database
[SQL Server]                             : Microsoft SQL database
[Microsoft][ODBC SQL Server Driver]      : Microsoft SQL database
[SQLServer JDBC Driver]                  : Microsoft SQL database
[SqlException                            : Microsoft SQL database
'80040e14'                               : Microsoft SQL database
mssql_query()                            : Microsoft SQL database
odbc_exec()                              : Microsoft SQL database
ORA-0                                    : Oracle database
ORA-1                                    : Oracle database
PostgreSQL query failed                  : PostgreSQL database
supplied argument is not a valid PostgreSQL result: PostgreSQL database
supplied argument is not a valid MySQL : MySQL database
mysql_fetch_array()                      : MySQL database
mysql_                                   : MySQL database
on MySQL result index                    : MySQL database
You have an error in your SQL syntax;  : MySQL database
MySQL server version for the right syntax to use: MySQL database
Incorrect syntax near                    : MySQL database
com.informix.jdbc                        : Informix database
Dynamic Page Generation Error:           : Informix database
Microsoft JET Database Engine error      : MS Access database
Microsoft OLE DB Provider for ODBC Drivers error : MS SQL database
[MySQL][ODBC                             : MySQL database
Column count doesn't match               : MySQL database
java.sql.SQLException                    : Java connector
<b>Warning</b>: ibase_                   : Interbase database
```

U slučajevima kada je rukovanje porukama o greškama pravilno izvedeno i kada se interne poruke ne prikazuju korisniku, ovakav pristup detekciji nije pogodan. W3af u tom slučaju pokušava detektirati SQL ubacivanje pomoću tehnike koja se naziva Slijepo SQL ubacivanje [20].

w3af konstruira dva tipa kôdova, jedan gdje se SQL upitu dodaje uvjet koji uvijek vraća istinu, i drugi gdje se dodaje uvjet koji vraća laž. Za oba tipa kôda postoje verzije s jednostrukim navodnicima, dvostrukim ili bez navodnika (ispis 2.24). W3af uspoređuje odgovore poslužitelja na oba upita i pokušava odrediti da li se razlikuju. Ako se odgovori razlikuju, to pokazuje da su dodatni uvjeti uspješno ubačeni u SQL upit, što znači da je došlo do SQL ubacivanja. Za usporedbu odgovora ostavljena je mogućnost odabira algoritma usporedbe.

Ispis 2.24: Primjeri kôda za SQL ubacivanje kod alata w3af

```
%i OR %i=%i
%i' OR '%i'='%i
%i" OR "%i"="%i
```

2.2.2. XPath ubacivanje

Ako se kao skladište podataka ne koristi SQL poslužitelj, već XML dokumenti, onda se SQL upiti zamjenjuju s XPath upitima. Ako ne postoji provjera korisnikovih podataka, može doći do XPath ubacivanja (engl. XPath injection) [22]. Princip ranjivosti i napada je jednak SQL ubacivanju. Ulazni podaci se koriste kod kreiranja upita bez provjere. Podaci mogu mijenjati

smisao upita i pristupati osjetljivim i skrivenim podacima u XML datoteci. Primjer jednog XPath upita je prikazan u ispisu 2.25.

Ispis 2.25: Primjer XPath upita

```
/users/user[username = 'xxx' and password = 'yyy']
```

Kada promijenimo vrijednost ulaznog podatka *password* tako da dodamo još jedan uvjet koji će uvijek vraćati istinu, zaobilazi se autentifikacija korisnika. Rezultat takvog XPath ubacivanja je prikazan u ispisu 2.26.

Ispis 2.26: Primjer XPath ubacivanja

```
/users/user[username = 'xxx' and password = 'yyy' or '1'='1']
```

Osim zaobilaženja autentifikacije, ranjivost XPath ubacivanja može poslužiti i za dohvaćanje sadržaja cijelog dokumenta. To se provodi kroz cijeli niz XPath upita, a tehnika se naziva slijepo XPath ubacivanje.

U nekim slučajevima, moguće je dohvatiti sadržaj cijelog XML dokumenta samo jednim XPath upitom. Napad se naziva XPath SQUAT i skraćenica je za engleski naziv XPath Single Query Attack. Do takvog napada može doći u radu AJAX aplikacija kod kojih se rezultat XPath upita vraća korisniku u originalnom XML obliku i tek tamo se pretvara u HTML kôd. Ispis 2.27. pokazuje kôd koji može poslužiti za dohvaćanje sadržaja cijelog dokumenta.

Ispis 2.27: Primjer XPath ubacivanja za dohvaćanje sadržaja cijelog XML dokumenta

```
'] | /* | /foo[bar='
```

Navodnik i uglata zagrada zatvaraju trenutni uvjet (*']*), zatim se koristi operator spajanja uvjeta (|) i dodaje se novi uvjet koji izabire sve elemente dokumenta počevši od početnog (/*) i na kraju se dodaje nevažan uvjet čija je svrha da otvori završan uvjet koji smo zatvorili u početku kôda.

Kao što je pokazano, XPath ubacivanje može dovesti do zaobilaženja procesa autentifikacije te krađe dijelova ili cijelog XML dokumenta koji sadrži povjerljive podatke.

Detekcija XPath ubacivanja je jednaka detekciji SQL ubacivanja. Skeneri ubacuju kôd koji bi trebao uzrokovati grešku na poslužitelju i onda se u odgovoru traži da li postoji poruka o grešci.

Wapiti i w3af koriste jednak kôd kao i kod SQL ubacivanja. Wapiti koristi kôd prikazan u ispisu 2.20, a kao popis poruka o greškama koristi jednu jedinu prikazanu u ispisu 2.28.

Ispis 2.28: Poruka o grešci koju Wapiti koristi kod detekcije XPath ubacivanja

```
XPathException
```

W3af posjeduje veću bazu znanja s učestalim porukama o greškama, a prikazana je u ispisu 2.29.

Ispis 2.29: Poruke o greškama koje koristi w3af kod detekcije XPath ubacivanja

```
Unknown error in XPath
org.apache.xpath.XPath
A closing bracket expected in
An operand in Union Expression does not produce a node-set
Cannot convert expression to a number
Document Axis does not allow any context Location Steps
Empty Path Expression
Empty Relative Location Path
Empty Union Expression
Expected ')' in
Expected node test or name specification after axis operator
Incompatible XPath key
Incorrect Variable Binding
libxml2 library function failed
libxml2
xmlsec library function
xmlsec
```

Kada se interne poruke o greškama ne prikazuju korisniku, može se primijeniti postupak detekcije ranjivosti koji se koristi kod slijepog SQL ubacivanja.

2.2.3. LDAP ubacivanje

LDAP (engl. Lightweight Directory Access Protocol) je široko raširen protokol namijenjen pristupu informacijama iz direktorija informacija. U slučajevima kada Web aplikacije upotrebljavaju korisnikove podatke da bi formirale LDAP upite i kada se ti podaci ne provjeravaju i filtriraju, može doći do LDAP ubacivanja (engl. LDAP injection) [23]. Moguće je da se vrijednosti parametara neke stranice automatski prosljeđuju LDAP poslužitelju. Tada URL ranjive stranice izgleda jednako kao i bilo koja druga stranica (2.30.).

Ispis 2.30: Primjer izgleda URL-a stranice ranjive na LDAP ubacivanje

```
http://www.someapp.com/ldap-queryuser.asp?name=xxx
```

Ako se vrijednost parametra *name* direktno koristi za konstruiranje LDAP upita, moguće je ubaciti dodatni LDAP kôd koji će promijeniti smisao upita i vratiti informacije koje ne bi trebale biti dostupne korisniku. Ispis 2.31 prikazuje primjere LDAP ubacivanja koji dohvaćaju korisnikovu lozinku te vraćaju imena svih korisnika sustava.

Ispis 2.31: Primjeri LDAP ubacivanja

```
http://www.someapp.com/ldap-queryuser.asp?name=xxx)(|password=*)
http://www.someapp.com/ldap-queryuser.asp?name=*
```

Ako je Web aplikacija ranjiva na LDAP ubacivanje, može napadaču pružiti dosta informacija o sustavu i korisnicima.

Detekcija LDAP ubacivanja se također provodi detekcijom učestalih poruka o greškama koje su uzrokovane nevaljanim korisničkim podacima/kôdom. Wapiti koristi jednak kôd kao i kod ostalih propusta ubacivanja prikazan u ispisu 2.20, a w3af za ovaj slučaj koristi posebne kôdove (ispis 2.32).

Ispis 2.32: Kôdovi za detekciju LDAP ubacivanja kod alata w3af

```
^(#$!@#$
)(()))
```

Oba alata traže jednu od slijedećih poruka koje su uzrokovane greškama u konstrukciji LDAP upita (ispis 2.33).

Ispis 2.33: Poruke o greškama karakteristične za LDAP ubacivanje

```
supplied argument is not a valid ldap
javax.naming.NameNotFoundException
```

2.2.4. CRLF ubacivanje

CRLF ubacivanje (engl. CRLF injection) je napad u kojem se dva specijalna znaka (Carriage Return i Line Feed) ubacuju u HTTP zaglavlje i time omogućuju dodavanje novih HTTP zaglavlja u odgovor Web poslužitelja. Do napada dolazi ukoliko ne postoji filtriranje specijalnih znakova.

Neke Web aplikacije uključuju korisnikove podatke u HTTP zaglavlja. Primjer u kojem je se koristi takav pristup je preusmjeravanje korisnika na novu stranicu u ovisnosti o odabranom jeziku stranice. Kôd koji obavlja tu funkciju prikazan je u ispisu 2.34.

Ispis 2.34: Primjer kôda za ubacivanje u HTTP zaglavlje

```
<% response.sendRedirect( "/lang.jsp?lang=" +
   request.getParameter("lang")); %>
```

Ako Web aplikacija ne filtrira vrijednost parametra *lang*, napadač može ubaciti znakove CR, LF i time kreirati dodatno HTTP zaglavlje u odgovoru Web aplikacije. Napadač to može iskoristiti za provođenje napada dijeljenja HTTP odgovora (engl. HTTP Response splitting) [24]. U ovom napadu, napadač napada ranjivu Web aplikaciju i ubacuje svoj kôd u HTTP zaglavlja u odgovoru. Kôd prekida trenutni odgovor i kreira potpuno novi dijeleći TCP tok na dva odgovora. Ukoliko napadač to napravi kroz žrtvin Web preglednik, drugi žrtvin zahtjev prema napadnutoj aplikaciji će umjesto zatražene stranice vratiti stranicu koju je napadač kreirao kroz prvi zahtjev. U kôd prikazan u ispisu 2.34 napadač može ubaciti maliciozan kôd koji može izgledati kao u ispisu 2.35.

Ispis 2.35: Primjer kôda za CRLF ubacivanje

```
foobar%0d%0aContent-Length:%200%0d%0a%0d%0a
HTTP/1.1%20200%20OK%0d%0a
Content-Type:%20text/html%0d%0a
Content-Length:%2027%0d%0a%0d%0a<html>Attackers HTML</html>
```

To će rezultirati TCP tokom prikazanom u ispisu 2.36.

Ispis 2.36: TCP tok uzrokovan CRLF ubacivanjem

```
HTTP/1.1 302 Moved Temporarily
Date: Wed, 24 Dec 2003 15:26:41 GMT
Location: http://somesite/lang.jsp?lang=foobar
Content-Length: 0

HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 27

<html>Attackers HTML</html>
Content-Type: text/html
... the rest of the original HTTP response headers and body
```

Drugi HTTP odgovor se vraća na korisnikov slijedeći zahtjev. Koristeći ovu tehniku, napadač može provesti razne tipove napada poput trovanja priručne memorije, promjene izgleda stranica, provođenja XSS napada,...

Detekcija CRLF ubacivanja je različita u odnosu na dosadašnje metode. Kako se u CRLF ubacivanju kôd ne ubacuje u tijelo odgovora, već u zaglavlja, koristi se drugačiji pristup. Svakom parametru koji se želi ispitati dodaju se dva posebne znaka: Carriage Return (ASCII kôd 13 - \r) i Line Feed (ASCII kôd 10 - \n) te novo zaglavlje nakon njih. U odgovoru se promatraju zaglavlja i traži se zaglavlje koje je ubačeno kroz kôd.

Wapiti koristi kôd prikazan u ispisu 2.37, dok w3af koristi kôd prikazan u ispisu 2.38.

Ispis 2.37: Kôd za detekciju CRLF ubacivanja

```
http://www.google.fr\r\nWapiti: version 1.1.6
```

Ispis 2.38: Kôd za detekciju CRLF ubacivanja

```
w3af\r\nVulnerable: Yes
```

W3af provjerava uspješnost ubacivanja malo detaljnije od Wapitija pa osim provjere imena novog zaglavlja provjerava i da li je vrijednost novog zaglavlja jednaka onoj koja je ubačena.

2.2.5. Ubacivanje OS naredbi

Ubacivanje OS naredbi (engl. OS commanding) je ranjivost u kojoj Web aplikacije prosljeđuju korisnikove podatke operacijskom sustavu koji ih koristi pri izvršavanju naredbi operacijskog sustava [25]. Do ranjivosti dolazi samo u slučaju kada Web aplikacija ne provjerava sadržaj korisnikovih podataka. Web aplikacija može koristiti korisnikove podatke kako bi pristupila datoteci koju je korisnik odredio (2.39).

Ispis 2.39: Primjer URL-a stranice ranjive na ubacivanje OS naredbi

```
http://www.someapp.com/cgi-bin/info.pl?name=John&template=tmp1.txt
```

Ako napadač promijeni vrijednost *template* parametra u kôd prikazan u ispisu 2.40, to bi moglo izlistati sadržaj direktorija i proslijediti ga naredbi *open* koja je namijenjena čitanju datoteke koju je korisnik trebao navesti.

Ispis 2.40: Primjer kôda kod ubacivanja OS naredbi

```
/bin/ls|
```

Ako Web aplikacija poziva *exec* naredbu i prosljeđuje joj korisnikove podatke (2.41), moguće je iskoristiti ju da izvrši dodatnu naredbu dodavanjem znaka ';' koji odvaja naredbe.

Ispis 2.41: Primjer korištenja exec naredbe

```
exec("ls -la $dir",$lines,$rc);
```

Izvršavajući naredbe na operacijskom sustavu, napadač može iskoristiti tu ranjivost da dohvati druge maliciozne programe na Web poslužitelj i pokrene nove tipove napada.

Prilikom detekcije ubacivanja naredbi operacijskog sustava, kao kôdovi za ubacivanje se koriste naredbe za prikaz nekih konfiguracijskih datoteka koje sigurno postoje na poslužitelju i znakovi koji služe za ulančavanje naredbi. Lista takvih kombinacija koju koristi w3af prikazana je u ispisu 2.42.

Ispis 2.42: Kôdovi za detekciju ubacivanja OS naredbi

```
/bin/cat /etc/passwd
&/bin/cat /etc/passwd
|/bin/cat /etc/passwd
;/bin/cat /etc/passwd
`/bin/cat /etc/passwd`
type C:\\boot.ini
&type C:\\boot.ini
|type C:\\boot.ini
;type C:\\boot.ini
```

Uz poruke o greški, u odgovorima se traže i neki dijelovi konfiguracijskih datoteka koji bi se trebali pojaviti ukoliko je došlo do izvršavanja ubačene naredbe. Popis koji koristi w3af prikazan je u ispisu 2.43.

Ispis 2.43: Dijelovi naredbi i poruke o greškama koje se traže prilikom detekcija ubacivanja OS naredbi

```
root:x:0:0:
daemon:x:1:1:
:/bin/bash
:/bin/sh
[boot loader]
default=multi(
[operating systems]
eval()'d code</b> on line <b>
Cannot execute a blank command in
Fatal error</b>:  preg_replace
```

2.2.6. SSI ubacivanje

SSI (engl. Server Side Include, SSI) [26] je jednostavan skriptni jezik koji se izvršava na poslužitelju, a najčešće se koristi za ubacivanje sadržaja jedne datoteke u neku drugu datoteku ili za izvršavanje operacijskih naredbi na sustavu. Općeniti oblik naredbi prikazan je u ispisu 2.44, dok su detalji jezika opisani u [27].

Ispis 2.44: Općeniti oblik SSI naredbe

```
<!-- #directive parameter=value parameter=value -->
```

Primjer korištenja najčešćih naredbi prikazan je u ispisu 2.45.

Ispis 2.45: Primjer korištenja najčešćih SSI naredbi

```
<!--#include virtual="header.html"-->
<!--#exec cgi="counter.cgi"-->
<!--#echo var="DATE_LOCAL" -->
```

Ako je Web poslužitelj podešen da izvršava SSI naredbe, napadač može ubaciti i izvršiti svoje naredbe. Primjer naredbi koje se mogu ubaciti prikazan je u 2.46.

Ispis 2.46: Primjer SSI naredbi

```
<!-- #include virtual="/web.config" -->
<!-- #exec cmd="/bin/ls /" -->
```

Napadači koriste SSI ubacivanje za izvršavanje svojih naredbi i pristup konfiguracijskim datotekama. Do ranjivosti dolazi samo u slučaju da nije ispravno izvedeno filtriranje korisnikovih podataka.

Detekcija SSI ubacivanja je slična detekcija ubacivanja OS naredbi. Kao kôdovi se ubacuju SSI naredbe i u odgovorima se traži dokaz da su one izvršene. W3af koristi dvije naredbe prikazane u ispisu 2.47, prvu za operacijski sustav Linux i drugu za operacijski sustav Microsoft Windows.

Ispis 2.47: SSI naredbe koje se koriste za detekciju SSI ubacivanja

```
<!--#include file=\"/etc/passwd\"-->
<!--#include file=\"C:\\boot.ini\"-->
```

Ukoliko se naredbe uspješno izvrše, jedna od dvije navedene konfiguracijske datoteke bi trebala biti uključena u stranicu pa se traže dijelovi te datoteke u odgovoru. Primjer dijelova konfiguracijskih datoteka koji bi uvijek trebali biti prisutni prikazan je u ispisu 2.48.

Ispis 2.48: Dijelovi konfiguracijskih datoteka koji se traže prilikom detekcije SSI ubacivanja

```
root:x:0:0:
daemon:x:1:1:
:/bin/bash
:/bin/sh
[boot loader]
default=multi(
[operating systems]
```

2.2.7. MX ubacivanje

MX ubacivanje (engl. MX injection) [28] je ranjivost koja je moguća kada Web aplikacije direktno komuniciraju s poslužiteljima za elektroničku poštu. Moguće je da su Web aplikacije ranjive na ubacivanje naredbi iz protokola za elektroničku poštu (IMAP i SMTP). Ako napadač uspije promijeniti IMAP ili SMTP komande koje se šalju poslužitelju za elektroničku poštu, dolazi do MX ubacivanja. To naravno ovisi o implementaciji filtriranja korisnikovih podataka. Napadači koriste ovaj tip napada da bi izvršavali nedopuštene akcije na poslužiteljima za elektroničku poštu, a to dovodi do odavanja informacija, izbjegavanja CAPTCHA sustava, slanja nepoželjne pošte i slično.

Detekcija MX ubacivanja je jednaka detekciji SQL ili XPath ubacivanja. Ubacuje se kôd koji treba uzrokovati grešku, a u odgovoru se traže uobičajene poruke o greškama. W3af prilikom detekcije MX ubacivanja traži poruke prikazane u ispisu 2.49.

Ispis 2.49: Poruke o greškama kod MX ubacivanja

```
Unexpected extra arguments to Select
Bad or malformed request
Could not access the following folders
To check for outside changes to the folder list go to the folders page
A000
A001
Invalid mailbox name
```

2.3. Izvođenje datoteka sa zlonamjernim sadržajem

Ranjivost izvođenja datoteka sa zlonamjernim sadržajem je još jedan tip ranjivosti koji se događa zbog nepravilnog filtriranja ulaznih podataka. Ako se korisnikovi podaci koriste za referenciranje određene datoteke, napadač može promijeniti te podatke kako bi učitao ili izvršio neku svoju zlonamjernu datoteku.

Iako je ovaj tip ranjivosti najzastupljeniji u programskom jeziku PHP, druge platforme poput Jave i .Net-a su također ranjive.

Najveći sigurnosni problem u PHP-u je upravo ranjivost udaljenog izvršavanja kôda (engl. Remote code execution) [29]. Najopasnije funkcije za ovaj tip ranjivosti su *include* i *require*, a koriste se za učitavanje dodatnog kôda u datoteku prije njenog izvršavanja. Primjer ranjivog kôda u PHP-u prikazan je u ispisu 2.50.

Ispis 2.50: Primjer ranjivog kôda u PHP-u

```
$report = $_POST['file'];
include $file;
```

Ako se parametar *file* ne provjerava, maliciozan kôd može biti učitan u datoteku i izvršen. Dio ostalih funkcija podložnih ovoj ranjivosti prikazan je u ispisu 2.51.

Ispis 2.51: Ostale PHP funkcije ranjive na izvršavanje kôda

```
fopen()
fsockopen()
popen()
system()
` (backtick operator)
eval()
include_once()
require_once()
```

U konfiguracijskoj datoteci *php.ini* postoji zastavica *allow_url_open* koja, kada je postavljena, omogućava čak učitavanje datoteka s drugih poslužitelja.

Ako ne postoji provjera promjene direktorija, moguće je pristupati datotekama iz ostalih direktorija (ispis 2.52.).

Ispis 2.52: Promjena direktorija kroz ulazne podatke

```
http://www.someapplication.com/execute.php?file=..\..\..\script.php
```

Napadači koriste ovu ranjivost da učitaju i izvrše maliciozan kôd na poslužitelju. Postoji dosta primjera iskorištavanja ovog tipa ranjivosti, a najčešće se pronalaze u PHP aplikacijama: forumima, blogovima, aplikacijama za upravljanje sadržajem i sl.

Iskorištavanje ove ranjivosti može dovesti do preuzimanja kontrole nad cijelim sustavom, a šteta koju može napraviti maliciozan kôd ovisi o pravima Web aplikacije na poslužitelju.

Automatski pristup problemu detekcije ovog tipa ranjivosti nije baš efikasan.

Skeneri ne mogu unaprijed znati koji parametar bi mogao bit podložan ovom tipu ranjivosti pa se svi ulazni parametri u aplikaciju moraju ispitati. Prilikom ispitivanja se pokušavaju referencirati poznate postojeće datoteke te se provjerava da li im je dozvoljen pristup.

2.4. Nesigurna izravna referenca na objekt

Kada su interni objekti referencirani koriteći URL, parametre forme ili identifikator sjednice i ne postoji ugrađena provjera prava pristupa određenom objektu, napadač može pristupiti drugim objektima kojima ne bi trebao imati pristup. Interni objekti mogu biti datoteke, direktoriji ili podaci u bazi podataka.

Pretpostavimo da se prikaz korisnikovih informacija prikazuje kroz Web stranicu čiji je URL prikazan u ispisu 2.53 i da ne postoji ugrađena provjera pristupa podacima. Napadač može promijeniti vrijednost parametra *id* i pristupiti informacijama nekog drugog korisnika. Provjera pristupa podacima bi trebala paziti da je korisnik autoriziran pristupiti samo svojim podacima.

Ispis 2.53: Primjer stranice za prikaz korisnikovih informacija

```
http://www.someapplication.com/viewuser.php?id=xxx
```

Također je česta ranjivost prilikom pristupa datotekama na poslužitelju. Ako se kao vrijednost ulaznog podatka očekuje ime datoteke koju treba ispisati, a ulazni podatak se ne provjerava na korištenje prečaca, moguće je pristupiti konfiguracijskim datotekama poslužitelja (ispis 2.54).

Ispis 2.54: Primjer korištenja prečaca

```
http://www.someapplication.com/viewfile.php?name=../../../../etc/passwd
```

Poznat je napad na Australski Taxation Office GST Start Up Assistance u 2000. godini kada je korisnik njihovog sustava otkrio da promjenom vrijednosti parametara u URL-u može pristupiti informacijama o drugim tvrtkama. Na taj način je prikupio informacije o 17.000 tvrtki i zatim ih sve obavijestio o ranjivosti [30].

Skeneri mogu uspješno detektirati ranjivost adresiranja direktorija (engl. path traversal), što također pripada u kategoriju nesigurnih referenci prema objektima. Ako postoji ugrađeno filtriranje i kontrola pristupa određenim objektima, i dalje postoji vjerojatnost da neki objekti nisu pravilno zaštićeni i da im napadač može pristupiti. Skener ne može znati koja pravila pristupa moraju biti ugrađena i ne može odlučiti da li nekom objektu treba biti omogućen pristup ili ne.

Kod detektiranja ranjivosti adresiranja direktorija, skeneri koriste putove prema poznatim konfiguracijskim datotekama i zatim u odgovoru provjeravaju da li je sadržaj tih datoteka uključen u odgovor. W3af koristi listu u ispisu 2.55 pri provjeri pristupa datotekama.

Ispis 2.55: Popis putova za ispitivanje ranjivosti kraćenja puta

```
../../../../../../../../etc/passwd
/etc/passwd
/etc/passwd\0
../../../../../../../../../boot.ini\0
C:\\boot.ini
C:\\boot.ini\0
```

Dijelovi konfiguracijskih datoteka za kojima se traga u odgovoru prikazani su u ispisu 2.56.

Ispis 2.56: Tekst korišten pri detekcije ranjivosti kraćenja puta

```
root:x:0:0:
daemon:x:1:1:
:/bin/bash
:/bin/sh
[boot loader]
default=multi(
[operating systems]
java.io.FileNotFoundException:
fread():
for inclusion (include_path=
Failed opening required
```

Pri pokušaju traženja referenci prema drugim tipovima objekata kao npr. ključevima u tablicama baze podataka, skeneri mogu mijenjati vrijednosti parametara, ali ne mogu znati da li bi korisnik trebao moći pristupiti tim objektima ili ne.

Skeneri mogu koristiti heuristiku za pogađanje čemu se smije pristupiti, ali to može rezultirati s puno pogrešnih procjena (engl. false positive).

2.5. Krivotvorenje zahtjeva

Krivotvorenje zahtjeva (engl. Cross Site Request Forgery, CSRF) je najraširenija ranjivost danas. Svaka Web aplikacija koja je ranjiva na XSS, automatski je ranjiva i na CSRF, ali odsutnost ranjivosti na XSS ne znači da aplikacija i dalje nije ranjiva na CSRF.

CSRF je napad u kojem će žrtva bez svojeg znanja slati zahtjeve nekom Web poslužitelju i izvršavati neželjene operacije. To se događa jer maliciozna ili ranjiva stranica koju je žrtva posjetila sadrži napadačev kôd koji od žrtvinog Web preglednika traži da provodi dodatne zahtjeve. Najjednostavniji primjer je prikazan u ispisu 2.57.

Ispis 2.57: Primjer neželjenog zahtjeva

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

Kada Web preglednik naiđe na dio HTML kôda prikazan u ispisu 2.57, on će automatski zatražiti stranicu navedenu u *src* parametru. Ako je korisnik do tada bio autentificiran na forum iz poveznice, sada će se odjaviti. Do toga dolazi jer se podaci za autentifikaciju na stranicu automatski šalju sa svakim zahtjevom. Iz poslužiteljeve točke gledišta, ovaj zahtjev je sasvim legalan i izgleda kao da je korisnik sam odlučio odjaviti se s aplikacije.

Odjava s nekog sustava je bezazlena akcija koja se može izvršiti. CSRF se koriti za razne maliciozne akcije: krađu financijskih sredstava, povjerljivih informacija i sl. Jedini uvjet

koji mora biti zadovoljen da bi napad bio uspješan je da je žrtva prijavljena na sustav koji se napada.

U Google-ovoj aplikaciji Google Docs [31] pronađena je CSRF ranjivost koja je omogućavala da se ukradu svi korisnikovi kontakti s Gmail-a, ukoliko je korisnik prijavljen [32]. Google Docs posjeduje skriptu koja definira povratnu funkciju koju je potrebno pozvati i prosljeđuje joj objekt sa svim korisnikovim kontaktima iz Gmaila. Skripta provjerava sjednički ključ, ali nije provjeravala s koje je stranice došao zahtjev. Napadač je mogao bilo gdje na Internetu umetnuti kôd koji bi zatražio od korisnikova Web preglednika da pozove ranjivu skriptu i onda dobivene podatke parsirati i pohraniti na željeno mjesto. Primjer kôda koji je dokazivao ranjivost je prikazan u ispisu 2.58. Konkretna ranjivost je ubrzano ispravljena ali je upozorila na opasnost curenja informacija kroz krivotvorenje zahtjeva.

Ispis 2.58: Primjer kôda za iskorištavanje CSRF ranjivosti na Google Docs aplikaciji

```
<script type="text/javascript">

function google(a){
    var emails;
    emails = "<ol>"
    for(i=0;i<a.Body.Contacts.length;i++){
        mails += "<li>"+a.Body.Contacts[i].Email+"</li>";
     }
 emails += "</ol>"
 document.write(emails);
}
</script>

<script src = "http://docs.google.com/data/contacts?
out=js&show=ALL&psort=Affinity&callback=google&max=99999"> </script>
```

Na dnu ispisa se poziva skripta koje referenicra *contacts* skriptu unutar Google Docs usluge. U parametru *callback* se definira funkcija koju je potrebno pozvati i prosljediti joj objekt sa svim korisnikovima kontaktima iz Gmaila. U prikazanom ispisu je to funkcija *google* čiji je kôd i prikazan na početku ispisa. Prikazan funkcija parsira dobiveni objekt i prikazuje informacije u obliku HTML kôda.

Vrlo je teško automatiziranim alatima za detekciju ranjivosti pronaći CSRF ranjivosti. U prirodi je napada da žrtva šalje legitimne zahtjeve poslužiteljima, ali bez svoga znanja. Problem je što skeneri ne mogu zaključiti da li je korisnik sam zatražio neki resurs ili je to napravio napadačev kôd ubačen u stranicu.

2.6. Ispuštanje informacija i neispravno rukovanje pogreškama

Ispuštanje informacija (engl. information leakage) i neispravno rukovanje pogreškama (engl. improper error handling) nije tip ranjivosti koji se može iskorištavati u dosadašnjem smislu te riječi. Ovaj tip ranjivosti ne omogućuje izvršavanje željenih akcija ili dohvat skrivenoga sadržaja, ali pruža informacije i pomaže u napadima koji to omogućuju. Ispuštanje informacija može otkriti informacije o operacijskom sustavu na kojem se nalazi Web poslužitelj, o verzijama aplikacija koje se tamo izvršavaju i njihovim sigurnosnim zakrpama. Također, informacije mogu otkriti imena i lokacije sigurnosnih kopija podataka, imena poslužitelja, IP adrese, korisnikova imena i lozinke. Takve informacije nalaze se u

zaboravljenim komentarima u kôdu, komentiranim dijelovima kôda ili na stranicama s greškom.

Automatsko izlistavanje direktorija je opcija na Web poslužitelju koja omogućava prikaz svih datoteka i direktorija unutar zatraženog direktorija ukoliko u njemu ne postoji definirana početna stranica. Uključivanje te opcije se također smatra ispuštanjem informacija.

Napadači mogu pokušati navesti aplikaciju da dođe do greške u njenom radu i da ona prikaže stranicu o grešci. Ako te stranice prikazuju previše informacija o grešci, mogu se koristiti za bolje shvaćanje rada aplikacije. Također mogu otkriti imena i putove do nekih internih objekata.

Jedan od čestih primjera je greška koja se prikazuje korisniku ako aplikacija pokuša kreirati nelegitiman SQL, XPath ili LDAP upit. Takve poruke o greškama često otkrivaju informacije o tipu i nazivu baze podataka, navode gdje se u upitu dogodila greška i mogu čak prikazati dio upita koji je problematičan. Sve te informacije mogu se koristiti za bolje shvaćanje internih upita i bolje provođenja napada ubacivanjem.

Skeneri imaju teškoće s razumijevanjem poruka o greškama, ali mogu biti podešeni da traže neke učestale poruke i prosuđuju da li je došlo do curenja informacija.

W3af pretražuje HTML kôd odgovora u potrazi za komentarima u kôdu koji odaju previše informacija. Pri tome se koristi listom prikazanom u ispisu 2.59 kao naznakom da se u komentaru pojavljuje nešto što korisnik ne bi trebao znati.

Ispis 2.59: Popis riječi koje w3af koristi prilikom potrage za curenjem informacija

```
'user', 'pass', 'xxx', 'fix', 'bug', 'broken', 'oops', 'hack', 'caution',
'todo', 'note', 'warning', '!!!', '???', 'shit'
```

Nadalje, neki skeneri traže informacije o privatnim IP adresama i SVN potpisima kao oblik curenja informacija.

2.7. Razbijena autentifikacija i kontrola sjednice

Razbijena autentifikacija se odnosi na neispravnu implementaciju mehanizama za autentifikaciju dok se kontrola sjednice odnosi na pripadajuće funkcije kao što su odjava, istek trajanja sjednice, tajna pitanja, ponovno postavljanje lozinke i sl.

Ako mehanizmi autentifikacije nisu dobro implementirani, moguće je iskoristiti tu slabost da bi se dobila veća prava nad aplikacijom. Čest je slučaj da proces autentifikacije vraća različite poruke ukoliko korisnik unese krivo korisničko ime, krivu lozinku ili oboje (2.60).

Ispis 2.60: Različite poruke kod krivo implementiranog sustava za autentifikaciju

```
Invalid username.
Invalid password for the given username.
Invalid username and password.
```

Ovakve poruke otkrivaju napadaču da li je pogodio korisnikovo ime, pa se on može prvo koncentrirati na pronalaženje postojećih korisničkih imena, a tek zatim na njihove lozinke. To je znatno brži proces od klasičnoga pogađanja kombinacije korisničkog imena i lozinke.

Neke aplikacije imaju skrivena područja koja ne koriste mehanizam autentifikacije i vjeruju da su sigurna od napadača jer nigdje ne postoji poveznica prema njima. Napadač može pogoditi nazive skrivenih stranica ili akcija i iskoristiti nedostatak autentifikacije.

Loš proces dobavljanja zaboravljene lozinke može se iskoristiti za dobavljanje ili mijenjanje korisnikove lozinke. Neki procesi obnove lozinke koriste tajno pitanje na koje korisnik treba odgovoriti, ali ne postoji zaštita od prevelikoga broja pokušaja pa napadač može isprobati vrlo veliki broj kombinacija i pogoditi pravi odgovor. Podsjetnici koji se nalaze uz takva pitanja mogu isto otkriti previše informacija.

Napadi na kontrolu sjednice se bave s autorizacijom korisnika. Korisnici se najčešće autoriziraju pomoću sjedničkog ključa koji se obično sprema u identifikatoru sjednice. Stoga, napadačev cilj su upravo sjednički ključevi.

Ako je algoritam za kreiranje sjedničkog ključa loš, napadač može pogoditi ili izračunati vrijednost drugog postojećeg sjedničkog ključa i pristupiti ciljanoj aplikaciji u ime korisnika čiji je sjednički ključ pronađen. Napad je poznat pod imenom predviđanje sjedničkog ključa (engl. credential/session prediction).

Jedan od napada na kontrolu sjednice je fiksacija sjedničkog ključa. U tom napadu, napadač preuzima legitiman sjednički ključ od aplikacije i šalje ga žrtvi te ju vara da ona iskoristi upravo taj ključ i pomoću njega se autentificira na aplikaciju. Napadač zatim dalje može koristiti taj sjednički ključ pristupajući aplikaciji s pravima žrtve. Predugo trajanje sjednice je ranjivost koja omogućuje napadačima da koriste stare sjedničke ključeve za autorizaciju.

Kao i kod nedovoljne autentifikacije, problem može biti i kod nedovoljne autorizacije kada aplikacija ne autorizira korisnika kada pristupa skrivenim stranicama ili akcijama.

Greške kod autentifikacije i kontrole sjednice su vrlo teške za automatsku detekciju. Skeneri ne mogu samostalno znati da li resurs kojem se pristupa treba biti zaštićen mehanizmima autentifikacije i autorizacije. Ako je implementirana autentifikacija, skeneri nailaze na poteškoće pri autentificiranju i najčešće trebaju pomoć korisnika alata (za upisivanje korisničkog imena i lozinke). Detekcija uspješnosti napada na kontrolu sjednice promjenom sjedničkog ključa je isto nejasna jer skener ne zna kada je napad uspješno izveden. Promjena sjedničkog ključa tako da odgovara drugom korisniku može rezultirati s vrlo sličnom ali i vrlo različitom stranicom. Dio gdje skeneri mogu puno pomoći je kod analize sakupljenih sjedničkih ključeva i proces predviđanja novih ključeva. Time se ispituje kompleksnost generiranja sjedničkih ključeva i eventualne mane u njemu. Ali i dalje ostaje problem odluke je li novogenerirani ključ ispravan.

2.8. Nesigurna kriptografska pohrana

Web aplikacije gotovo uvijek imaju potrebu za korištenjem i pohranom povjerljivih informacija. Ako postoji potreba za autentifikacijom i autorizacijom korisnika, onda postoji i potreba da se podaci potrebni za taj proces sigurno pohrane. Za sigurnu pohranu potrebno je koristiti algoritme za enkripciju, ali ponekad ni to nije ispravno izvedeno.

Najgori oblik pohrane povjerljivih podataka je pohrana u obliku čistog teksta. Ako napadač uspije dobiti pristup bazi podataka ili datotekama gdje su podaci pohranjeni, korisnička imena i lozinke bit će odmah otkriveni.

Spremanje lozinki u obliku sažetka (engl. hash) isto nije neprobojna zaštita. Napadač može koristiti Rainbow tablice [33] da pronađe originalnu vrijednost za otkriveni sažetak.

Rainbow tablice su unaprijed generirane tablice koje sadrže sve kombinacije teksta i sažetka za određenu grupu znakova i određenu dužinu teksta. Proračun takve tablice za sve vrijednosti bi zahtijevao previše vremena i prostora, ali generiranje takvih tablica za neke česte vrijednosti lozinki se može izvesti u prihvatljivom vremenu. Jednom kada je tablica generirana, napadač je može iskoristiti da brzo potraži originalnu vrijednost nekog sažetka.

Najsigurniji način pohrane lozinki je korištenje jakog enkripcijskog algoritma za sažetak u kombinaciji s miješanjem lozinke i neke proizvoljne vrijednosti (engl. salt) kako bi se napadač prisilio da iznova računa Rainbow tablice ako želi primijeniti taj pristup.

Alati za detekciju ranjivosti ne mogu znati da li je korištena enkripcija prilikom pohranjivanja podataka. Gledajući na Web aplikaciju izvana, nije moguće odrediti kako se podaci spremaju i da li postoje neke mane. Čak ni pomoć korisnika prilikom analize ne može pomoći pri određivanju ovog tipa ranjivosti. Jedini način detekcije je inspekcija kôda na poslužitelju te baze podataka ako se podaci spremaju u bazu ili nekog drugog tipa spremanja podataka.

2.9. Nesigurna komunikacija

Problem kod nesigurne komunikacije nastaje kada se osjetljivi podaci (poput korisnikove lozinke) prenose kroz računalnu mrežu nezaštićeni algoritmima za enkripciju i provjeru integriteta. Prisluškivanje računalne mreže (žičane ili bežične) može otkriti pakete koji sadrže lozinke i ako oni nisu enkriptirani, jednostavno izložiti lozinke napadaču. Korištenje slabih enkripcijskih algoritama ili algoritama u kojima su pronađeni propusti se jednako tako smatra ranjivošću.

Skeneri mogu jednostavno provjeriti da li je korišten siguran protokol pri prijenosu povjerljivih informacija između klijenta i poslužitelja, ali zato nikako ne mogu znati da li se i kako ti podaci prenose između glavnog i ostalih poslužitelja u sustavu. Najjednostavniji pristup detekciji je provjera koji se protokol koristi za prijenos informacija (HTTP ili HTTPS) ili koji se pristup koristi (80, 443 ili neki drugi).

W3af vrši provjeru mogućnosti pristupa zaštićenim resursima preko nezaštićenog protokola. Za svaku stranicu za koju nađe da joj se pristupa preko sigurnog protokola (HTTPS), alat pokušava zatražiti istu stranicu i preko nezaštićenog protokola (HTTP).

2.10. Neuspješna zaštita pristupa URL-u

Ukoliko ne postoji provjera kontrole pristupa URL-u, napadač može pogoditi skrivene lokacije ili stranice i pristupiti im zaobilazeći autentifikaciju. Ne postojanje poveznice prema skrivenim resursima ne osigurava njihovu tajnost. Napadač može nasumce pogađati imena Web stranica ili akcija u tehnici koja se na engleskom naziva *forced browsing*.

Alati za detekciju ne mogu znati da li koji resurs treba biti zaštićen ili bi svaki korisnik trebao biti u mogućnosti pristupiti mu.

Skeneri pokušavaju naći skrivene resurse ali i dalje ne znaju da li bi trebala postojati zaštita pristupa tim resursima. Za pronalazak skrivenih resursa koriste se razne tehnike: pogađanje imena direktorija i datoteka, korištenje poznatih i čestih imena i sl.

W3af uzima ime svake stranice i pokušava dohvatiti istu stranicu mijenjajući ekstenzije dokumenta u nadi da će pronaći zaboravljene, stare ili sigurnosne kopije dokumenta.

Prilikom pogađanja vrijednosti parametara, w3af koristi servis Google Sets [34] koji na temelju nekoliko poznatih vrijednosti predlaže nove vrijednosti iz istog skupa. Na primjer, za pronađene stranice prikazane u ispisu 2.61 i vrijednosti parametra *color*, Google Sets će vratiti novi skup mogućih vrijednosti koji će sadržavati: *black, white, green*.

Ispis 2.61: Primjer pronađenih stranica i vrijednosti parametra

```
http://www.someapplication.com/index.asp?color=blue
http://www.someapplication.com/index.asp?color=red
```

Dodatni primjer pogađanja resursa kod w3af alata je mijenjanje brojčanih vrijednosti koje se pojavljuju u vrijednostima parametara ili imenima stranica. W3af pokušava dohvatiti isti resurs s tim da brojčanu oznaku poveća ili smanji za jedan.

3. Detekcija ranjivosti pomoću algoritama za raspoznavanje uzoraka

U prethodnom poglavlju pokazane su najkritičnije ranjivosti koje se javljaju kod Web aplikacija te je pokazano kako postojeći alati za provjeru sigurnosti detektiraju te ranjivosti. Pokazano je da detekcija najvećeg broja ranjivosti (izvršavanje napadačkog kôda i propusti ubacivanja) ovisi o detekciji učestalih grešaka koje se pojavljuju u odgovorima poslužitelja ili o detekciji kôda koji je ubačen kroz ulazne parametre aplikacije. Ukoliko je poslužitelj podešen da odgovara sa specijalno prilagođenim i općenitim stranicama o greškama ili uopće ne vraća stranice s greškom takav pristup može propustiti detekciju pojedinih ranjivosti.

Početna ideja rada je bila da se detektiraju svi takvi slučajevi u kojima Web aplikacija vraća specijalno prilagođenu stranicu s greškom koja se ne može detektirati koristeći detekciju učestalih poruka o greškama. Ideja se temeljila na algoritmima ra zaspoznavanje uzoraka koji bi grupirale dobivene stranice prema njihovom izgledu i izdvojila slučajeve u kojima je došlo do greške na Web poslužitelju. Daljnom analizom tematike utvrđeno je da sustav ne može samostalno odrediti koja stranica opisuje grešku pa je sustav modificiran da samo grupira različite stranice, a korinik će vizualnom analizom pojedinih primjeraka iz grupa odrediti koja grupa sadrži valjane stranice, a koja grupa stranice s greškama.

Za pretpostaviti je da će poslužitelj na većinu zahtjeva odgovoriti ispravno, vraćajući korisniku zatražene stranice ili resurse. No ako se za vrijednosti nekog parametra unese vrijednost koja će izazvati grešku na poslužitelju, odgovor na takav zahtjev bi se trebao razlikovati od prethodnih zahtjeva koji nisu uzrokovali grešku. Cilj alata izrađenog u sklopu ovog rada je sakupljati sve odgovore i izdvajati ih u posebne grupe prema sličnosti stanica te odrediti u kojim se situacijama aplikacija ponaša nepredvidivo. Za takve slučajeve se onda ostavlja korisniku da provede detaljniju analizu potencijalne ranjivosti. Na taj način se iz velikog spektra podataka koji se skupljaju od aplikacije korisniku pruža samo manji, interesantniji podskup koji predstavlja slučajeve u kojima bi moglo doći do greške, odnosno, ranjivosti.

Sustav bi trebao biti u stanju detektirati slučajeve u kojima može doći do ranjivosti izvršavanja napadačkog kôda, propusta ubacivanja ili drugog nepredvidivog ponašanja Web aplikacije i Web poslužitelja.

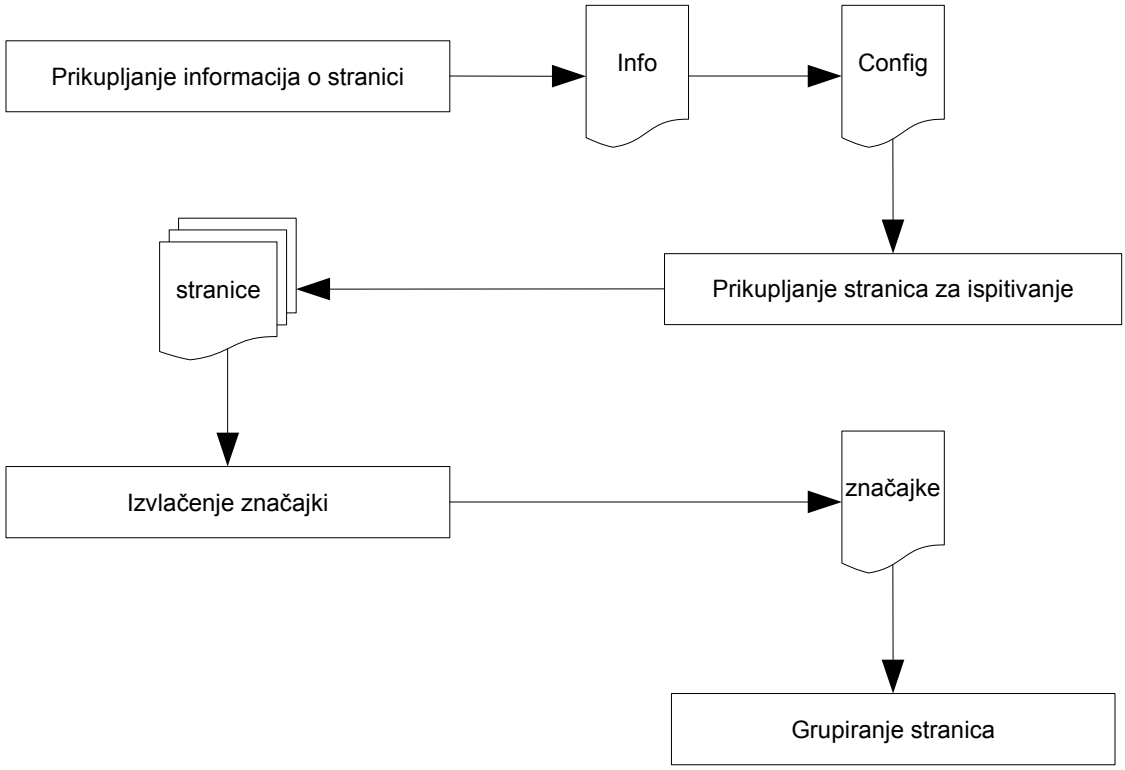
3.1. Dizajn sustava

Proces detekcija potencijalnih ranjivosti je podijeljen na četiri procesa:

- prikupljanje poveznica i parametara Web aplikacije
- prikupljanje stranica za ispitivanje
- analiza stranica i izvlačenje bitnih značajki
- grupiranje prema značajkama stranica

Sustav je dizajniran na način da svaki proces bude implementiran jednim modulom. Na taj način se ostavlja korisniku mogućnost da svaki od koraka izvodi zasebno i podešava parametre modula.

Komunikacija između modula će se vršiti preko međudatoteka. Na taj način će korisnici imati uvid u rezultate svakog koraka i ujedno će ih koristiti kao ulazne podatke za slijedeći dio procesa. Stroga definicija međudatoteka će osigurati da moduli rade jednoznačno. Model cjelokupnog sustava prikazan je na slici 3.1.



Slika 3.1: Model sustava

Elementi sustava će biti sada posebno opisani. Uz svaki proces bit će opisane i ulazne datoteke koja se očekuju kao ulazni parametar.

3.1.1. Prikupljanje poveznica i parametara

Za uspješnu detekciju sigurnosnih ranjivosti kod Web aplikacija potrebno je ispitati sve ulazne parametre u aplikaciju. Prikupljanje ulaznih parametara se izvodi u procesu koji se često naziva *crawling* ili *spidering*. To je proces u kojem sustav kreće od početne stranice Web aplikacije i traži na njoj sve poveznice i parametre. Sustav dalje rekurzivno pretražuje nove poveznice i zapisuje informacije o novim stranicama i njihovi parametrima.

Unutar našeg rada, odlučeno je promatrati samo poveznice koje se nalaze direktno ugrađene u HTML kôd stranica. Primjer poveznica koje se moraju detektirati prikazan je u ispisu 3.1. Poveznice koje se stvaraju dinamički, izvršavanjem JavaScript kôda te poveznice koje se dohvaćaju putem AJAX zahtjeva se neće tražiti jer kompleksnost takvog sustava nadilazi okvir ovog rada.

Ispis 3.1: Primjer poveznice ugrađene u HTML kôd

```
<a href="index.php">home</a>
```

Zamišljeno je i da se ostvari način na koji će se korisnik moći prije ili prilikom procesa prikupljanja poveznica autentificirati na Web aplikaciju. To je od posebne važnosti zbog toga što bez autentifikacije veliki broj stranica ostaje nedostupan korisniku.

Sve prikupljene informacije se spremaju u XML datoteku sa strogo definiranom strukturom. Na taj način, bilo koji postojeći ili novoizgrađeni modul može koristiti sakupljene informacije.

3.1.2. Prikupljanje stranica za ispitivanje

Jednom kada imamo podatke o pronađenim stranicama i njihovim parametrima potrebno je odrediti koji se sve parametri žele ispitivati te s kojim ispitnim vrijednostima želimo vršiti ispitivanje. Odlučeno je da je konfiguraciju modula za ispitivanje najbolje izvesti također kroz XML datoteku u kojoj je moguće definirati testove za sve parametre odjednom ili svakom parametru odrediti svoj skup vrijednosti i dodatne opcije koje se odnose samo na njega. Konfiguracijski dokument je osmišljen kao nadogradnja na dokument dobiven od prethodnog modula. Problem koji je poznat i nije riješen je kompleksnost odabira i konfiguracije testiranja za aplikacije s velikim brojem stranica i parametara. Ručno mijenjanje teksta u velikim XML dokumentima nije najsretnije riješenje, ali je osmišljena metoda koje bi trebala proces učiniti jednostavnijim. Dokument sa svim informacijama o aplikaciji će se moći automatski nadograditi u konfiguracijski dokument, a od korisnika će se tražiti samo sitne preinake oko podešavanja testova. Za budući razvoj sustava ostavljena je ideja kreiranja grafičkog sučelja za određivanje parametara za ispitivanje i za rukovanje skupovima s ispitnim vrijednostima.

3.1.3. Analiza i izvlačenje značajki stranice

Prilikom dizajniranja sustava posebno je obraćena pažnja na dizajn procesa izvlačenja značajki iz dohvaćenih stranica. Zaključeno je da sustav mora biti dovoljno modularan da dodavanje nove značajke koja se želi promatrati ne utječe niti na jedan modul te da nije potrebna nikakva promjena u implementaciji modula. Okvir u kojem je dizajniran ovaj dio procesa je dizajniran na način da se učitavanje modula za izvlačenje značajki odvija dinamički i sustavu nije bitno koliko ih ima i kakve operacije se provode nad stranicama. Jedina potreba za pisanjem novog kôda je dio unutar novog modula koja definira kako se izračunava nova značajka. U formatu stranica koje su ulazni podaci u ovaj proces, predviđen je dio za razmjenu dodatnih informacija o načinu na koji je stranica dohvaćena.

3.1.4. Grupiranje stranica

Za odabir formata zapisa datoteke koja je ulazni podatak u ovaj proces ponovno je odabran format koji je maksimalno fleksibilan i omogućuje korištenje informacija sa širokim spektrom algoritama za raspoznavanje uzoraka. Format tablice sa značajkama je preuzet od Orange sustava [35]. Orange sustav omogućuje razne operacije nad ulaznim podacima, odabir velikog broja algoritama za klasifikaciju i grupiranje te mogućnost prikaza rezultata u grafičkom sučelju. Odabir upravo njihovog formata zapisa tablice značajki ostavlja korisniku brojne opcije. Implementacija algoritma za grupiranje je također preuzeta iz Orange sustava jer zadovoljava većinu potreba sustava.

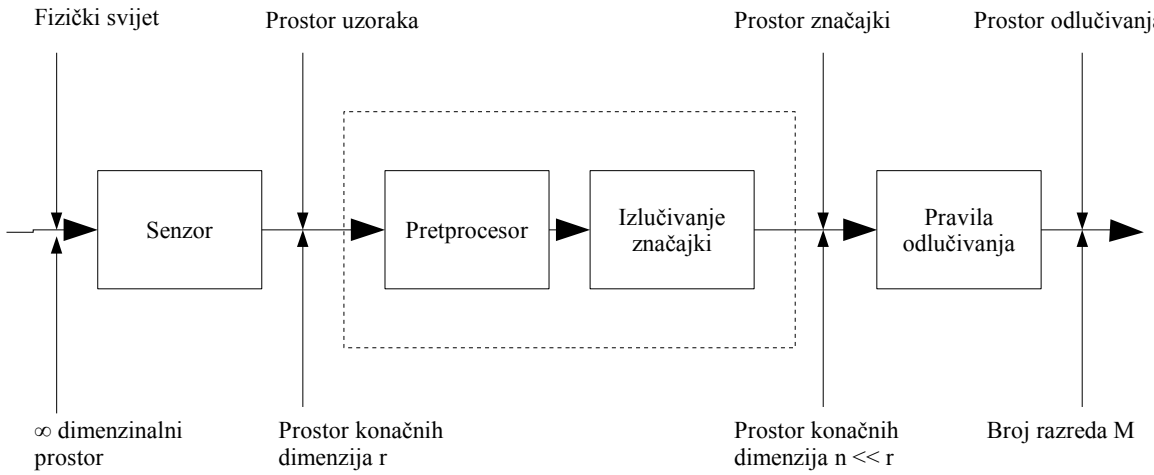
U slijedećem poglavlju će biti dan uvod u teoriju raspoznavanja uzoraka i bit će opisan algoritam K srednjih vrijednosti koji se koristi u sustavu.

3.2. Algoritmi za raspoznavanje uzoraka

Algoritmi za raspoznavanje uzoraka se koriste u vrlo širokom spektru primjena. Koriste se za raspoznavanje vizualnih uzoraka (pisma, znakova, otisaka prstiju, elemenata na fotografijama,...), zvučnih uzoraka (govora, jezika, zvuka strojeva,...), biomedicinskih uzoraka (EKG, bolesti,...) te za raspoznavanje ponašanja kompleksnih sustava (npr. vremenska prognoza). Samo područje raspoznavanja uzoraka je blisko povezano s umjetnom inteligencijom, a teorija raspoznavanja uzoraka se temelji na brojnim heurističkim, jezičnim i matematičkim metodama.

Heuristički pristup je temeljen na ljudskom iskustvu i uvelike ovisi o iskustvu kreatora sustava. Jezični se pristup zasniva na opisu osnovnih elemenata sustava i odnosa među njima koji je opisan nekim formalnim jezikom. Takvi klasifikatori su zapravo analizatori jezičnog niza, a uzorci se razvrstavaju u razrede koji odgovaraju pojedinim gramatikama. Gramatike uzoraka se sastoje od skupa elemenata, konstanti i pravila i opisuju formalni jezik. Matematički pristup koristi brojne matematičke metode: klasične operacije, matematičku logiki, *fuzzy* skupove. Možemo ga podijeliti na deterministički i stohastički, ovisno o upotrebi statističkih svojstava uzoraka i razreda. U daljnjem tekstu će biti razmatran matematički pristup.

Problem raspoznavanja uzoraka se obično dijeli na tri dijela, odnosno prostora. Prostor uzoraka je reprezentacija objekta iz stvarnog svijeta u računalu. Objekt u stvarnom svijetu posjeduje beskonačno puno značajki dok je u prostoru uzoraka predstavljen s konačnim brojem. Putem senzora se analogne informacije diskretiziraju i stvara se reprezentacija objekta u digitalnom svijetu. Broj značajki takvog uzorka, tj. dimenzionalnost prostora je i dalje vrlo velika. Procesom izlučivanja značajki se prostor uzoraka znatno smanjuje. Izdvajaju se samo bitne značajke uzoraka koje će se koristiti u pravilima odlučivanja, te se mogu vršiti razne radnje filtriranja i obrade značajki. Novodobiveni prostor se naziva prostor značajki. Uzorci se iz prostora značajki pomoću kriterija i pravila odlučivanja dijele u M razreda koji čine prostor odlučivanja. Sva tri prostora su prikazana na slici 3.2 koja prikazuje model sustava za raspoznavanje uzoraka.



Slika 3.2: Model sustava za raspoznavanje uzoraka

Uzorci su predstavnici objekata koje treba razvrstati, a sastoje se od odabranih svojstava promatranog objekta. Za razlikovanje dvaju objekata nije potrebno promatrati sva svojstva, već karakteristični podskup koji dovoljno dobro opisuje sam objekt. U našem sustavu, opis objekta u prostoru uzoraka je HTML kôd dohvaćene Web stranice, a prostor značajki sadrži samo značajke koje smo izlučili iz HTML kôda. Prostor odlučivanja se sastoji od razreda u koje se razvrstavaju dohvaćene stranice.

Razredi su skupovi uzoraka sa zajedničkim svojstvima. Cilj ovog rada je razvrstati Web stranice u razrede u kojima će se nalaziti međusobno slične stranice. Sličnost stranice se promatra u odnosu na korisnikovo shvaćanje izgleda stranice. Pri tom se ne promatra i interpretira sam tekst koji se nalazi na stranici već se gleda kako je taj tekst posložen, uspoređuju se međusobni odnosi elemenata na stranici i sl.

U radu sustava je potrebna metoda koja je u stanju grupirati uzorke prema njihovoj sličnosti bez prethodnog znanja o broju grupa i karakteristikama uzoraka. Zbog toga su od posebnog značaja interesantni postupci raspoznavanja grupa koji se još nazivaju i učenjem bez učitelja. Za razliku od razvrstavanja uzoraka u već poznate grupe gdje su poznati i karakteristični predstavnici grupa, postupci za raspoznavanje grupa ne traže takvo znanje. Grupiranje definiramo kao određivanje razreda uzoraka unutar početnog skupa uzoraka u kojem će se nalaziti međusobno slični uzorci.

Sličnost uzoraka možemo definirati na različite načine, ali najintuitivnija mjera sličnosti je udaljenost dvaju uzoraka. Za *mjeru udaljenosti* opet postoje razne funkcije ali jedino nužno svojstvo je da funkcija zadovoljava slijedeće uvijete:

1. $D(\vec{x}_k, \vec{x}_l) = 0 \quad za \quad \vec{x}_k = \vec{x}_l,$
   $D(\vec{x}_k, \vec{x}_l) \neq 0 \quad za \quad \vec{x}_k \neq \vec{x}_l,$
2. $D(\vec{x}_k, \vec{x}_l) = D(\vec{x}_l, \vec{x}_k),$
3. $D(\vec{x}_k, \vec{x}_l) \leq D(\vec{x}_k, \vec{x}_j) + D(\vec{x}_j, \vec{x}_l)$

Najpoznatija funkcija udaljenosti je Euklidova udaljenost koja udaljenost dvaju uzoraka definira na slijedeći način:

$$D = \| \vec{x}_k - \vec{x}_l \| = \sqrt{\sum_{i=1}^{n} (x_{ki} - x_{li})^2}$$

Euklidova udaljenost je pogodna za primjere kada nam je dovoljna invarijantnost grupa s obzirom na translaciju i rotaciju uzoraka. Linearne transformacije nad uzorcima utječu na grupiranje uzoraka ukoliko se koristi ova funkcija udaljenosti. Drugi primjer funkcije udaljenosti koja se koristi je Mahalanobisova udaljenost čija formula glasi:

$$D = (\vec{x} - \vec{m})^T C^{-1} (\vec{x} - \vec{m})$$

C je kovarijantna matrica, vektor m je središnja vrijednost, a vektor x je proizvoljni uzorak. Mahalanobisova udaljenost je normirana te je zbog toga invarijantna na linearne transformacije. U našem radu će se koristiti Manhattan udaljenost koja se definira na slijedeći način:

$$D = \| \vec{x}_k - \vec{x}_l \| = \sum_{i=1}^{n} |x_{ki} - k_{li}|$$

Osim mjerenja sličnosti pomoću funkcija udaljenosti, moguće je koristiti i nemetričke mjere. Jedna od njih je mjera kosinusa kuta koja dva uzorka smatra sličnima ako je njihovo usmjerenje s obzirom na ishodište slično. Takva mjera sličnosti je neovisna o rotiranju ili rastezanju prostora uzoraka, ali je osjetljiva na pomak i linearne transformacije nad uzorcima. Formula mjere kosinusa kuta je:

$$s(\vec{x}_k, \vec{x}_l) = \frac{\vec{x}_k^{\,T} \cdot \vec{x}_l}{\|\vec{x}_k\| \cdot \|\vec{x}_l\|}$$

Drugi primjer nemetričke mjere je Tanimotova mjera koja je inačica binarnog oblika mjere. Upotrebljava se u biološkoj taksonomiji a glasi:

$$s(\vec{x}_k, \vec{x}_l) = \frac{\vec{x}_k^{\,T} \vec{x}_l}{\vec{x}_k^{\,T} \vec{x}_k + \vec{x}_l^{\,T} \vec{x}_l - \vec{x}_k^{\,T} \vec{x}_l}$$

Uz odabir mjere sličnosti potrebno je odrediti i *kriterij grupiranja* uzoraka. Uzorci se mogu grupirati raznim postupcima: heurističkim postupcima, postupcima temeljenim na teoriji grafova, postupcima nalaženja ekstrema kriterijske funkcije. Jedna od najčešće upotrebljavanih kriterijskih funkcija jest zbroj kvadrata odstupanja između uzoraka u grupi i aritmetičke sredine grupe:

$$J = \sum_{j=1}^{N_c} \sum_{\vec{x} \in S_j} \|\vec{x} - \vec{m}_j\|^2$$

N_c je broj grupa, S_j je j-ta grupa uzoraka, vektor x opisuje uzorak iz grupe uzoraka a vektor m_j opisuje središte grupe.

Algoritam grupiranja koji se koristi u našem sustavu je algoritam K srednjih vrijednosti (MacQueen, 1967.). Algoritam uvjetuje točno određen broj grupa K u koje će se uzorci grupirati. Predstavnici grupa se mogu odrediti proizvoljno jer će se tijekom rada algoritma oni mijenjati pa inicijalne vrijednosti nisu toliko bitne. Poželjno je da predstavnici uzoraka ne budu preblizu jedan drugom. Zatim će se ostali uzorci grupirati prema udaljenosti od predstavnika i potom se računaju novi predstavnici grupa. Algoritam se zasniva na minimizaciji kriterijske funkcije:

$$J = \sum_{j=1}^{N_c} \sum_{\vec{x} \in S_j} \|\vec{x} - \vec{m}_j\|^2$$

Cijeli algoritam se može prikazati u četiri koraka:

1. Odabir K predstavnika grupa. Predstavnici grupa se odabiru proizvoljno i jedini je uvjet da broj grupa K mora biti manji ili jednak broju uzoraka N. Predstavnike grupa u prvom koraku označavamo kao:

$$\vec{m}_1(1), \vec{m}_2(1), \ldots, \vec{m}_K(1)$$

2. Svi preostali uzorci se dodjeljuju u grupe tako da se uzorak dodjeljuje grupi čiji mu je predstavnik najbliži. Korak se ponavlja kroz k iteracija a u k-toj iteraciji se uzorci grupiraju pomoću relacije:

$$\vec{x} \in S_j(k), \text{ ako je } |\vec{x} - \vec{m}_j(k)| < |\vec{x} - \vec{m}_i(k)| \quad \forall i = 1, 2, \ldots, K \quad ; \quad i \neq j$$

3. Računaju se novi predstavnici grupa. Predstavnici grupa će biti središta novonastalih grupa. Nova središta grupa u k-tom koraku označavamo sa:

$$\vec{m}_j(k+1), \quad j = 1, 2, \ldots, K$$

a računaju se tako da kriterijska relacija

$$J = \sum_{j=1}^{K} \sum_{\vec{x} \in S_j(k)} \|\vec{x} - \vec{m}_j(k+1)\|^2$$

bude minimalna. Novi predstavnici, odnosno , sredine uzoraka koji minimiziraju tu relaciju su aritmetičke sredine uzoraka svake grupe:

$$\vec{m}_j(k+1) = \frac{1}{N_j} \sum_{\vec{x} \in S_j(k)} \vec{x} \quad za \quad j = 1, 2, \ldots, K \quad i \quad N_j \text{ je broj uzoraka u grupi } S_j(k)$$

4. Koraci 2. i 3. se ponavljaju sve dok se predstavnici grupa ne prestanu mijenjati, odnosno, dok ne vrijedi:

$$\vec{m}_j(k+1) = \vec{m}_j(k) \quad \forall \quad j = 1, 2, \ldots, K$$

4. Sustav za detekciju ranjivosti kod Web aplikacija

Sustav za detekciju ranjivosti je realiziran po uzoru na dizajn opisan u prethodnom poglavlju. Posebna je pažnja posvećena prenosivosti sustava na razne operacijske sustave te na modularnu izgradnju. Modularnost sustava osigurava jednostavnu zamjenu pojedinih modula boljima te mogućnost korištenja rezultata ili međurezultata u kombinaciji s novim modulima. Dijelovi sustava za koje je postojala zadovoljavajuća implementacija su iskorišteni u cijelosti dok su se pojedini dijelovi koji su djelomično zadovoljavali zahtjeve sustava nadogradili te prilagodili sustavu.

4.1. Korištene tehnologije

4.1.1. Python

Čitav sustav je izgrađen u programskom jeziku Python [36]. Python je programski jezik visokog nivoa, prvi put objavljen 1991. godine. Jezik se temelji na otvorenom razvojnom modelu baziranom na radu zajednice, a nadgledan je od strane neprofitne organizacije Python Software Foundation. Iako razni dijelovi jezika imaju formalnu specifikaciju i standarde, jezik u cijelosti nije formalno specificiran. Najčešći standard koji se koristi je Cpython implementacija, koja je napisana u programskom jeziku C i distribuira se za razne operacijske sustave: Microsoft Windows, većina Linux/UNIX sustava, Mac OS. U Pythonu je moguće pisati kôd u nekoliko paradigmi. Najčešće korištene paradigme su funkcijska, objektno-orijentirana i imperativna, ali je moguće koristiti i druge. To ostavlja mogućnost korisniku da koristi paradigmu koja mu najviše odgovara ili da kombinira različite paradigme kada je to potrebno. Još neke od snaga Pythona su korištenje sustava dinamičkog određivanja tipa parametra te automatsko rukovanje memorijom.

Uz veliku standardnu biblioteku, Python posjeduje i veliku zbirku dodatnih paketa (engl. packages) koje razvija Python zajednica, a trenutna brojka prelazi 3300 paketa [37]. Paketi su pogodni za brzu realizaciju već postojećih i riješenih problema i lako se ugrađuju u postojeće projekte.

Za ostvarenje našeg sustava korišten je Python verzije 2.5 te zbog nekih specifičnosti rada sustava nije podržana kompatibilnost s ranijim verzijama. Uz sam Python, korišteni su i neki postojeći paketi koji će ovdje ukratko biti nabrojani i opisani:

1. PyXML

   PyXML paket [38] je skupina biblioteka za parsiranje XML dokumenata uz pomoć programskoj jezika Python. Unutar našeg sustava ga koristimo za kreiranje, čitanje i pisanje XML dokumenata te za provođenje XPath upita nad XML dokumentima.

2. µTidylib (opcionalno)

   µTidylib paket [39] je paket koji je zapravo samo ovojnica za Python oko biblioteke HTML Tidy, a služi za popravljanje HTML kôda koji ne odgovara propisima HTML jezika. Jednom kada se uz pomoć ovakvog modula HTML kôd ispravi, moguće je na temelju njega kreirati DOM objekt i lakše pristupati elementima stranice. U našem sustavu se koristi u modulu *Crawler* prilikom analize stranice i potrage za novim poveznicama i parametrima. Za sam rad sustava paket nije nužan, ali je poželjan.

3. BeautifulSoup (opcionalno)

   Paket BeautifulSoup [40] služi za parsiranje HTML/XML sadržaja, a pri tome može rukovati i neispravno formiranim kôdom. U našem sustavu ga koristimo u iste svrhe kao i paket µTidylib. Neispravan kôd se prvo prosljeđuje ovom paketu i ako nakon ispravaka kôd i dalje ne odgovara propisima te ne može se pravilno parsirati, šalje se paketu µTidylib da on pokuša ispraviti greške. Jednako kao i µTidylib, paket nije nužan, ali je poželjan.

4. ClientCookie (opcionalno)

   Paket ClientCookie [41] je paket za rukovanje s kolačićima na klijentskoj strani aplikacije. Paket se pokazao korisnim jer omogućuje rukovanje s kolačićima koji se postavljaju kroz *meta* elemente u HTML kôdu. Sustav može funkcionirati i bez ovog paketa, ali onda neće moći rukovati kolačićima koji se postavljaju na taj način pa će u određenim situacijama biti zakinut za proces autentifikacije na Web aplikaciju.

4.1.2. Orange

Orange [35] je sustav za otkrivanje znanja u skupovima podataka koji uključuje niz tehnika za pretprocesiranje, modeliranje i istraživanje podataka. Temeljen je na komponentama razvijenim u programskom jeziku C++ koje se mogu pozivati direktno (rijetko se upotrebljava), kroz Python skripte ili kroz objekte u grafičkom sučelju (engl. Orange widgets).

Orange je realiziran kao okvir temeljen na komponentama pa omogućuje korištenje različitih postojećih komponenti ali i ugradnju novih. Posjeduje neke elementarne komponente i brojne kompleksne koje su izgrađene nad elementarnim. Neke od trenutnih mogućnosti Orange sustava uključuju:

- Fleksibilnost ulaznih i izlaznih podataka

  Orange sustav je u mogućnosti čitati i pisati u dokumente raznih formata. Neki od češće korištenih su dokumenti u kojima se navode podaci odvojeni tabulatorima ili zarezima, ali jednako tako je moguće koristiti i C4.5 format koji se koristi u Quinlanovom C4.5 sustavu [42]. U operacijskom sustavu MS Windows moguće je podatke čitati i iz Excel formata datoteke. Korisnicima se ostavlja i mogućnost pisanja vlastitih funkcija za čitanje i pisanje podataka te njihova integracija u Orange sustav.

- Pretprocesiranje podataka

  Velike su mogućnosti pretprocesiranja podataka prije nego se oni upotrijebe u jednom od odabranih algoritama. Nad ulaznim podacima je moguće vršiti selekciju pojedinih atributa i selekciju pojedinih uzoraka. Također je moguće vršiti selekciju nad podacima kojima nedostaju vrijednosti za pojedine atribute ili čije vrijednosti nisu bitne. Jednako tako je moguće dodavanje nepoznatih vrijednosti, odnosno zamjena poznatih vrijednosti nepoznatima za potrebe testiranja klasifikatora. Podacima je moguće dodavati i šum, pridruživati težine pojedinim atributima, pretvarati kontinuirane podatke u diskretne te izbaciti identične uzorke koji se ponavljaju.

- Odabir algoritma za klasifikaciju ili grupiranje uzoraka

  Orange sustav posjeduje već implementiran velik broj algoritama za klasifikaciju i grupiranje uzorka. Iz skupine algoritama za klasifikaciju moguće je izabrati

klasifikacijska stabla, Bayesov klasifikator, k-NN (k najbližih susjeda), SVM, CN2 i dr. Također je moguće koristiti i Breimanove algoritam pakiranja (engl. bagging) i algoritam Random Forest. Implementiran je i algoritam punjenja (engl. boosting) čiji su autori Freund i Schapire. Za grupiranje uzoraka moguće je koristiti algoritme hijerarhijskog grupiranja, algoritam K srednjih vrijednosti, višedimenzionalno dijeljenje (MDS) i dr.

- Metode za opis podataka i valjanost odabranih tehnika

  Za vizualizaciju podataka moguće je koristiti grafičko sučelje koje posjeduje razne widget kontrole za rad s podacima i algoritmima te za prikaz rezultata. Implementirane su i razne metode odabira podataka za učenje i testiranje klasifikatora te metode za valjanost klasifikatora i usporedbu rezultata među više klasifikatora.

U radu je korištena posljednja stabilna verzija Orange sustava (verzija 0.9.66) zajedno sa svim potrebnim modulima koji se distribuiraju uz njen instalacijski paket. Uz sam Orange sustav, korišten je i dodatni modul koji implementira nekoliko algoritama za grupiranje uzoraka [43]. Iz tog modula je odabran algoritam K srednjih vrijednosti i on je dalje korišten u radu za grupiranje uzoraka.

4.1.3. WSAT crawler

Za potrebe sakupljanja informacija o aplikaciji odlučeno je koristiti postojeći modul iz sustava za analizu sigurnosti Web aplikacija pod imenom WSAT. Testiranjem i analizom postojećeg modula *crawler* primijećeni su određeni nedostaci:

1. Nemogućnost rukovanja kolačićima postavljanim kroz *meta* elemente HTML-a

   Prilikom korištenja postojećeg *crawler* modula otkriveno je da je on u mogućnosti rukovati samo s kolačićima koji se postavljaju kroz HTTP zaglavlja zahtjeva i odgovora. Prilikom korištenja modula su pronađene stranice koje se oslanjaju na postavljanje kolačića kroz *meta* elemente u HTML kôdu pa u tim slučajevima proces autentifikacije na Web aplikaciju nije uspijevao.

2. Nedostatak informacija o parametrima formi

   Razvijanjem sustava pokazala se potreba za dodatnim informacijama o parametrima formi. Postojeći sustav je navodio samo ime parametra i njegovu vrijednost pa je bilo potrebno implementirati mehanizam koji opisuje parametar sa svim potrebnim svojstvima.

3. Greška u modulu HTMLParser

   Za parsiranje stranice prilikom potrage za novim poveznicama se koristi Pythonov modul *HTMLParser.HTMLParser*. Pokazalo se da parser ima grešku u slučaju kada se analizira dio kôda koji se nalazi u *script* elementu. Ako se unutar *script* elementa pojavi unutar nekog teksta neispravno zatvoreni HTML element, modul bi se prestao izvršavati javljajući poruku o greški.

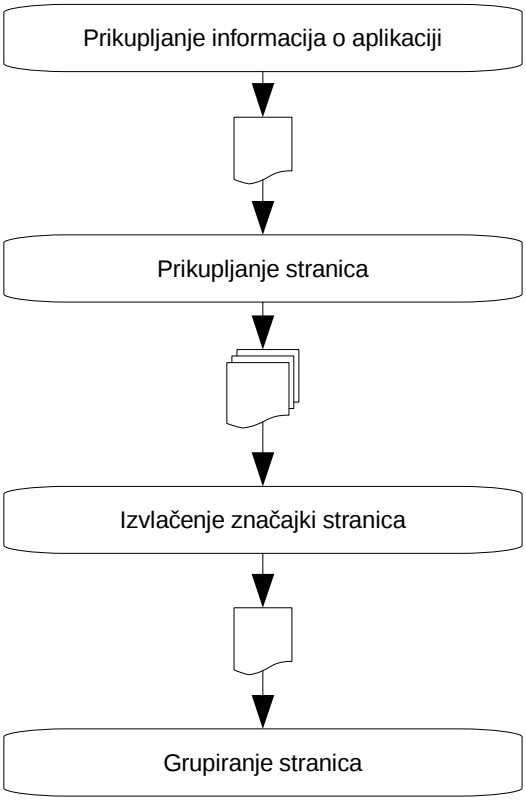
4. Greška prilikom kreiranja apsolutnih poveznica

   Korištenjem sustava uočena je greška prilikom kreiranja apsolutnih poveznica. Ukoliko je na Web stranici bio definiran *base* element koji određuje početnu vrijednost na koju se nadovezuju relativne poveznice, modul to nije bio u stanju

prepoznati. To je rezultiralo neispravnim kreiranjem apsoultnih poveznica i ulaskom modula u beskonačnu petlju prilikom dohvaćanja novih stranica.

4.2. Faze obrade

Sustav detekcije ranjivosti je podijeljen u četiri faze. Svaka faza je implementirana kao zasebni modul, a sustav se može koristiti koristeći svaki modul zasebno ili ulančavajući pojedine module što će biti objašnjeno u poglavlju o korištenju sustava. Moduli komuniciraju preko međudatoteka koje će biti objašnjene u slijedećem poglavlju. Proces detekcije potencijalne ranjivosti prikazan je na slici 4.1.



Slika 4.1: Četiri koraka u detekciji potencijalne ranjivosti

4.2.1. Prikupljanje informacija o aplikaciji

Modul za prikupljanje informacija o aplikaciji je preuzet iz sustava WSAT (engl. Web Security Assessment Tool). WSAT je projekt čiji je cilj razviti sustav za ispitivanje sigurnosti Web aplikacija. Pri tom je jedan dio projekta orijentiran na ispitivanje poznatih aplikacija za koje je poznata strukutra poveznica i formi te postoji njen opis. Drugi dio projekta je posvećen ispitivanju aplikacija o kojima se ne posjeduje nikakvo znanje. U prvoj fazi rada na projektu napravljen je sustav za identifikaciju poznatih aplikacija koje se izvršavaju na Web poslužitelju i unutar toga je razvijen modul za prikupljanje informacija o svim poveznicama i ulaznim parametrima testirane aplikacije.

Prilikom korištenja tog modula pri izradi ovog rada otkrivene su neke mane i greške u modulu pa su one u implementaciji ispravljene i ovdje objašnjene. Uz to su dodane i neke dodatne mogućnosti kako bi modul mogao efikasnije funkcionirati.

1. Promjene u arhitekturi modula

   Najbitnija promjena u arhitekturi modula je uvođenje novih objekata koji opisuju poveznice i njihove parametre. Upotrebom novih objekata olakšano je korištenje i manipulacija poveznicama.

2. Ispravak greške u modulu *HTMLParser*

   Ispravljena je greška u Pythonovom modulu *HTMLParser.HTMLParser* koja nije omogučavala pravilno parsiranje kôda unutar *script* elementa.

3. Ugrađivanje ClientCookie paketa za kontrolu rukovanja kolačićima

   Ugrađivanje ClientCookie paketa je rješilo nedostatak prilikom rukovanja kolačićima. Koristeći ovaj paket omogućeno je rukovanje kolačićima koji se postavljaju kroz meta elemente HTML kôda pa je na stranicama koje korsite taj pristup omogućen proces autentifikacije na Web aplikaciju.

4. Ispravak greške prilikom kreiranja apsolutnih poveznica.

   U postojeći parser HTML kôda ugrađena je opcija koja detektira pojavljivanje base elementa i omogućuje ispravno kreiranje apsolutnih poveznica.

5. Prikupljanje dodatnih informacija o stranicama

   Osim informacija o poveznicama i formama koje se pronalaze u stranicama, dodana je još mogućnost traženja skrivenih stranica po direktorijima koji se nalaze u pronađenim poveznicama. Direktoriji se uzimaju iz svih pronađenih poveznica na stranici. Uz to je dodana i sposobnost prikupljanja dodatnih informacija o parametrima formi. Tako sadašnji modul osim samog imena forme, prikuplja i informacije o tipovima parametara.

6. Mogućnost korištenja posebno specificiranih separatora parametara.

   U modul je ugrađena mogućnost parsiranja poveznica i parametara stranica koje ne koriste uobičajene znakove za razdvajanje parametara u URL-u, već koriste posebno specificirane znakove ili nizove znakove. Jedan od takvih primjera prikazan je u ispisu 4.1 a prikazuje URL jedne od stranica poznatog servisa eBay.

Ispis 4.1: primjer URL-a na servisu eBay

```
http://books.listings.ebay.com/Accessories_Other_W0QQfromZR4QQsacatZ45115QQsoc
mdZListingItemList
```

Vidimo da ova stranica koristi nestandardni oblik razdvajanja parametara i njihovih vrijednosti, a klasični separatori su zamijenjeni novima prema ispisu tablici 4.1.

Tablica 4.1: Zamjena separatora kod servisa eBay

'?'	_W0QQ
'&'	QQ
'='	Z

Sustav sada koristi mehanizam koji omogućava redefiniranje separatora pa je u modulu *crawler* bilo potrebno napraviti potrebne izmjene.

Sama logika rada modula je ostala jednaka. Parsiranje HTML kôda također nije mijenjano osim što je ispravljena spomenuta greška i što je dodana mogućnost prikupljanja dodatnih informacija. Modul i dalje posjeduje komunikacijski dio koji je odgovoran za komunikaciju s Web aplikacijom. Komunikacija se ostvaruje korištenjem nekoliko komunikacijskih rukovatelja (engl. handlers):

- HTTP rukovatelj (rukovatelj koji služi za komunikaciju HTTP protokolom)
- HTTPS rukovatelj (rukovatelj koji služi za komunikaciju HTTPS protokolom)
- Proxy rukovatelj

  Ukoliko je potrebno promet između klijenta i poslužitelja propuštati kroz posrednika, moguće je definirati proxy poslužitelj s kojim će ovaj rukovatelj znati funkcionirati

Uz komunikacijske rukovatelje postoje i rukovatelji za kolačiće. U ovisnosti o prisutnim paketima odabire se jedan od rukovatelja:

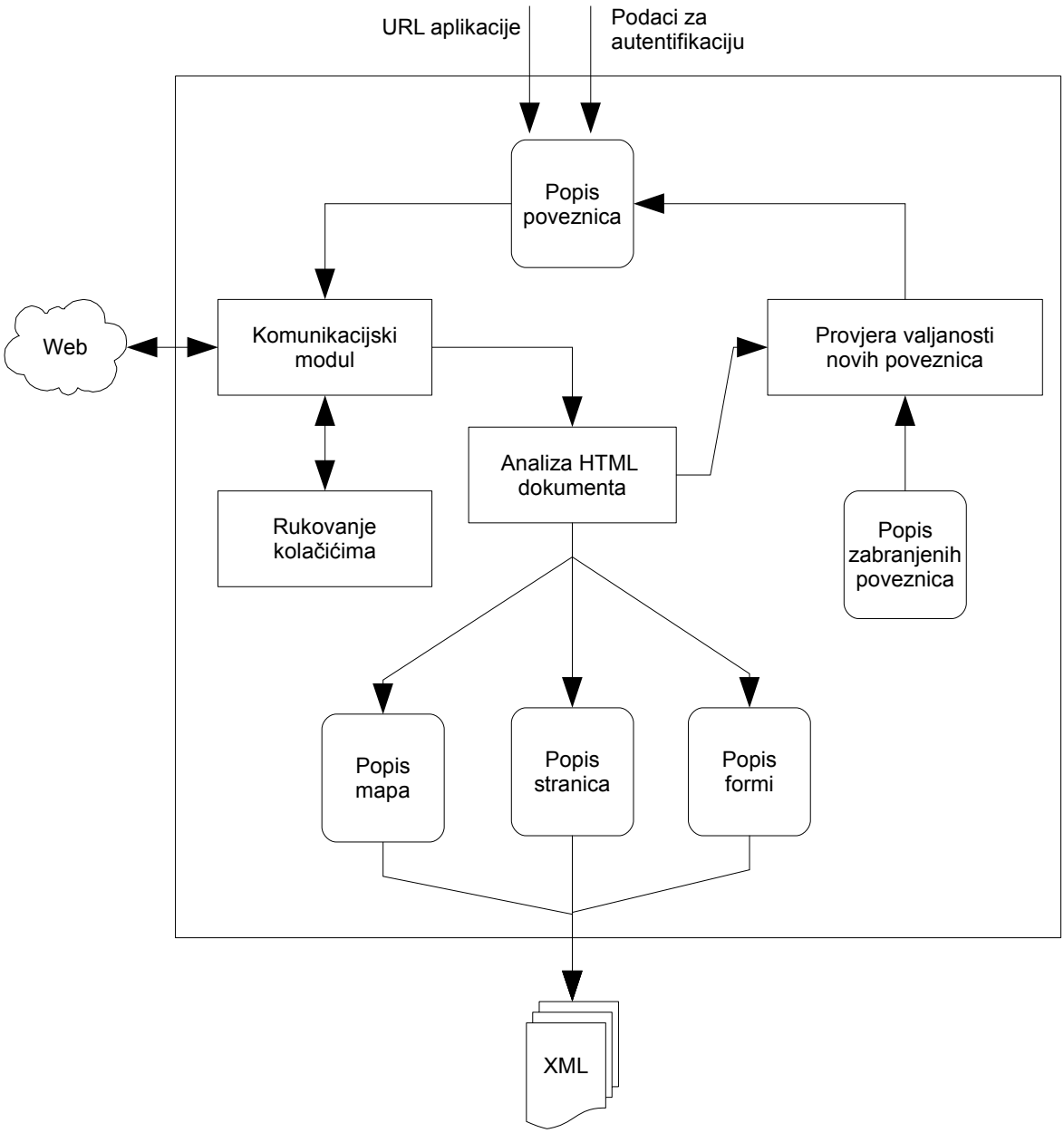
- Osnovni rukovatelj kolačićima

  Osnovni rukovatelj kolačićima je u stanju čitati, postavljati i brisati sjedničke kolačiće koji se postavljaju kroz zaglavlja HTTP zahtijeva, odnosno odgovora.

- Napredni rukovatelj kolačićima.

  Ako je moguće u sustav uklopiti već spomenuti paket *ClientCookie*, onda će se on automatski uključiti u rad. Napredni rukovatelj kolačićima uz rukovanje preko HTTP zaglavlja omogućuje i rukovanje preko *meta* elemenata u HTML kôdu.

Logički rad modula se nije znatno promijenio u odnosu na osnovnu verziju. Prije samog pokretanja rada modula, definira se URL Web aplikacije koja će se promatrati te se definiraju opcionalni parametri. Kroz opcionalne parametre je moguće postaviti kolačić (engl. cookie) u kojem se nalazi sjednički identifikator ili potrebni podaci za autentifikaciju na stranicu. Ukoliko korisnik ne posjeduje kolačić moguće je aktivirati mehanizam za interaktivnu autentifikaciju. Također je moguće definirati poveznice koje modul ne smije slijediti i analizirati. Uz to postoji mogućnost određivanja dubine puta koju modul treba slijediti i razine informacija koja se ispisuje na ekran. Interna arhitektura je prikazana na slici 4.2.

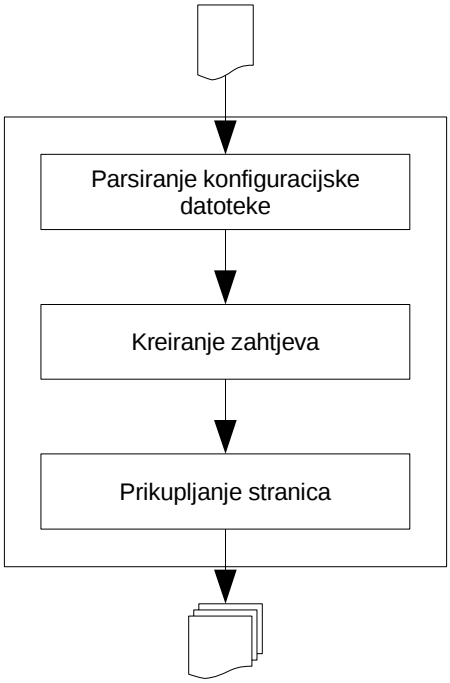


Slika 4.2: Proces prikupljana informacija o aplikaciji

Modul posjeduje popis poveznica s putovima kako je modul došao do njih. Svaka od tih poveznica se dohvaća preko komunikacijskog dijela modula i vrši se analiza dohvaćenog HTML kôda. Kroz analizu se prikupljaju informacije o dosad nepoznatim poveznicama, formama ili direktorijima, a za sve prikupljene poveznice se provjerava da li zadovoljavaju pravila valjanosti (da li je dopušteno slijediti poveznicu i da li je maksimalna dubina puta premašena). Kada se lista poveznica za analizu isprazni, modul je završio s radom. Svi rezultati se pretvaraju u strogo definiran XML oblik i spremaju u XML datoteku. Uz datoteku s prikupljenim informacijama o aplikaciji sprema se i posebna datoteka u kojoj su definirani kolačići koji su prikupljeni u radu.

4.2.2. Prikupljanje stranica

Modul za prikupljanje stranica, nazvan *fuzzer*, izvršava drugu fazu procesa detekcije ranjivosti prikazanu u ispisu 4.1. Modul je razdvojen u nekoliko dijelova rada, a uvid u rad modula je prikazan na slici 4.3. Modulom se upravlja pomoću konfiguracijske datoteke opisane kasnije u poglavlju, a izlaz iz modula su prikupljene stranice u formatu koji je također posebno definiran.



Slika 4.3: Proces prikupljanja stranica

Prvi dio procesa prikupljanja stranica je određivanje stranica koje treba dohvatiti te parametara koje treba ispitati. To se vrši parsiranjem konfiguracijske datoteke. Moguće je definirati da se ispitaju svi parametri svih pronađenih poveznica ili je moguće odabrati samo neke i vršiti ispitivanje nad njima. Parametrima za ispitivanje također treba pridružiti vrijednosti s kojima će se vršiti testiranje. Svi ti podaci se učitavaju iz konfiguracijske datoteke, a primjer odabira samo jednog parametra za ispitivanje je prikazan u ispisu 4.2. Masnim slovima su otisnute vrijednosti za podešavanje rada modula dok je ostatak elemenata dio dokumenta dobiven od *crawler* modula. Detaljan opis oba dokumenta dan je u idućem poglavlju.

Ispis 4.2: Primjer stranice određene za ispitivanje u konfiguracijskoj datoteci

```
...
<form id="26" fuzz:type="onebyone">
  <url>http://testphp.acunetix.com/search.php?test=query</url>
  <data>searchFor=on&amp;goButton=go</data>
  <from>http://testphp.acunetix.com</from>
  <parameters>
    <param id="3" method="get" name="test" fuzz:max="50" fuzz:set="2">
      <values>
        <value>query</value>
      </values>
    </param>
    <param id="1" method="post" name="searchFor" type="text">
      <values>
        <value>on</value>
      </values>
    </param>
    <param id="2" method="post" name="goButton" type="submit">
      <values>
        <value>go</value>
      </values>
    </param>
  </parameters>
</form>
...
```

Nakon što modul parsira sve potrebne podatke iz konfiguracijske datoteke, generira zahtjeve koji će se slati Web aplikaciji. U ovom procesu se spajaju parametri s njihovim vrijednostima, a međusobna interakcija je podržana na tri različita načina:

- jedan po jedan (ključna riječ *onebyone*)

  Ukoliko postoje dva parametra koji se ispituju, kreiranje zahtjeva za ispitivanje se vrši na način da se prvo prvom parametru dodjeljuju vrijednosti iz njegovog pripadajućeg skupa dok se drugi parametar ne mijenja. Zatim se mijenjaju vrijednosti drugog parametra uz početnu vrijednost prvog. Za početne vrijednosti se uzimaju vrijednosti pronađene uz parametre u poveznicama ili formama, a ukoliko ne postoji početna vrijednost koristi se prazan tekst. Neka imamo dva parametra p_1 i p_2 s početnim vrijednostima *1* i *2* te neka je svakom parametru pridružen skup vrijednosti:

  $$s_1 \in A_1, B_1, C_1, \ldots \quad \textit{skup vrijednosti za parametar } p_1$$
  $$s_2 \in A_2, B_2, C_2, \ldots \quad \textit{skup vrijednosti za parametar } p_2$$

  Uz takve postavke generirat će se zahtjevi koji će imati parametre kao u koloni *onebyone* na slici 4.4.

- spajanje u parovima (ključna riječ *pairwise*)

  Ako je kombiniranje parametara definirano kao spajanje u parovima (*pairwise*), onda se vrijednosti iz svih pridijeljenih skupova koriste paralelno. Maksimalni broj zahtjeva jednak je broju vrijednosti u skupu s minimalnim brojem elemenata. Za jednake parametre i skupove iz prošlog primjera, spajanje će biti izvedeno kao u koloni *pairwise* na slici 4.4.

- svaki sa svakim (ključna riječ *cross*)

  Treći oblik spajanja je spajanje sa svim mogućim kombinacijama. U tom se slučaju uz prvu vrijednost iz skupa s_1 koriste sve vrijednosti iz skupa s_2. Zatim se to sve ponavlja za sve vrijednosti iz skupa s_1. Generirane kombinacije su prikazane u koloni *cross* na slici 4.4.

onebyone	*pairwise*	*cross*
p_1 p_2	p_1 p_2	p_1 p_2
A_1 2	A_1 A_2	A_1 A_2
B_1 2	B_1 B_2	B_1 B_2
C_1 2	C_1 C_2	C_1 C_2
...	...	...
1 A_2		A_1 A_2
1 B_2		B_1 B_2
1 C_2		C_1 C_2
...		...

Slika 4.4: Načini spajanja parametara i vrijednosti

Jednom kada su kreirane sve mogućnosti spajanja parametara s ispitnim vrijednostima, kreiraju se zahtjevi za dohvaćanje stranica i zatim kreće faza prikupljanja stranica. Stranice se pohranjuju u niz direktorija koji se dinamički kreira tako da se svaka stranica koja se ispituje nalazi u svojem direktoriju koji ima ime kao stranica, a svi direktoriji sa stranicama se nalaze u posebnom direktoriju koji je imenovan kao Web aplikacija. Ako npr. ispitujemo dvije stranice čiji su URL-ovi prikazani u ispisu 4.3, onda će se unutar definiranog direktorija za pohranu rezultata kreirati direkotriji prikazani u ispisu 4.4 i u njih će se spremiti dohvaćene stranice.

Ispis 4.3: Primjer URL-ova ispitivanih stranica

```
http://www.posluzitelj.com/index.asp
http://www.posluzitelj.com/page.asp
```

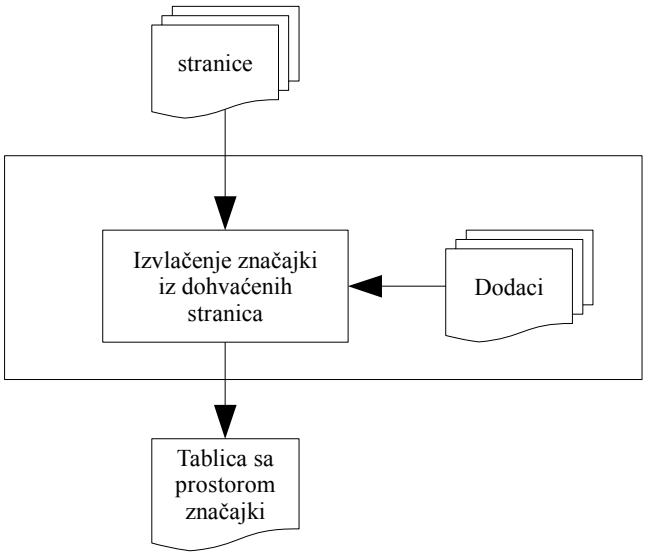
Ispis 4.4: Primjer direkotrija u koje se spremaju dohvaćene stranice

```
./www.posluzitelj.com/index.asp/
./www.posluzitelj.com/page.asp/
```

Na taj način se dohvaćene stranice za svaku pojedinu stranicu spremaju u svoj direktorij i doprinose preglednosti rezultata.

4.2.3. Izvlačenje značajki stranica

Modul za izvlačenje značajki stranica je napravljen kao proširivi okvir. Okvir rukuje s posebnim dodacima, a svaki je dodatak zadužen za izvlačenje jednog određenog podatka iz stranice. Niz takvih dodataka stvara vektor koji opisuje karakteristike te stranice, a niz takvih vektora formira matricu koja opisuje sve značajke dohvaćenih stranica. Ta matrica se zatim prosljeđuje sustavu za grupiranje stranica koji razvrstava stranice prema njihovoj sličnosti. Proces je jednostavan i prikazan je na slici 4.5.



Slika 4.5: Proces izvlačenja značajki iz dohvaćenih stranica

Okvir je implementiran tako da kao ulaz prima put do direktorija u kojem se nalaze dohvaćene stranice te iz posebnog direktorija učitava dodatke za izvlačenje značajki. Svaki dodatak izvlači iz stranice jednu značajku stranice, a niz svih dodataka kreira vektor značajki koji opisuje stranicu. Dodavanje novih atributa u prostor značajki se svodi na ubacivanje novog dodatka u pripadajući direktorij.

Dodaci moraju poštivati određena pravila kako bi okvir znao pravilno rukovati s njima. Dinamički se učitavaju prilikom pokretanja sustava i ubacivanje novih dodataka ne uzrokuje nikakve potrebe za promjenama u samom okviru. Dodaci moraju poštovati slijedeća pravila:

- Svi dodaci se moraju nalaziti u direktoriju *plugins* koja se nalazi u istom direktoriju s modulom za izvlačenje značajki. Okvir iz tog direktorija učitava sve datoteke s ekstenzijom *.py* izuzev datoteke *__init__.py* i dinamički kreira objekte za rad.
- Ime klase koja se nalazi u datoteci mora biti jednako imenu datoteke. Na taj način okvir jednoznačno određuje kako se zove klasa unutar dodatka i u stanju je instancirati objekt.
- Svaka klasa mora imati tri atributa prikazana u tablici 4.2.

Tablica 4.2: Popis obaveznih atributa u klasi

Ime atributa	**Opis atributa**
`attributeName`	Atribut opisuje ime atributa u prostoru značajki
`attributeDescription`	Atribut opisuje čemu služi taj atribut u prostoru značajki
`attributeType`	Opisuje tip atributa, može biti *discrete* ili *continuous*

- Svaka klasa mora imati definiranu metodu *ExtractDataFromString* koja kao ulaz prima jedan parametar tekstualnoga tipa u kojem se nalazi HTML kôd stranice, a kao izlaz vraća podatak brojčanog ili tekstualnog tipa koji sadrži dobivenu vrijednost značajke.

Primjer dodatka koji za određenu stranicu vraća broj pojavljivanja *img* elementa u HTML kôdu je prikazan u ispisu 4.5.

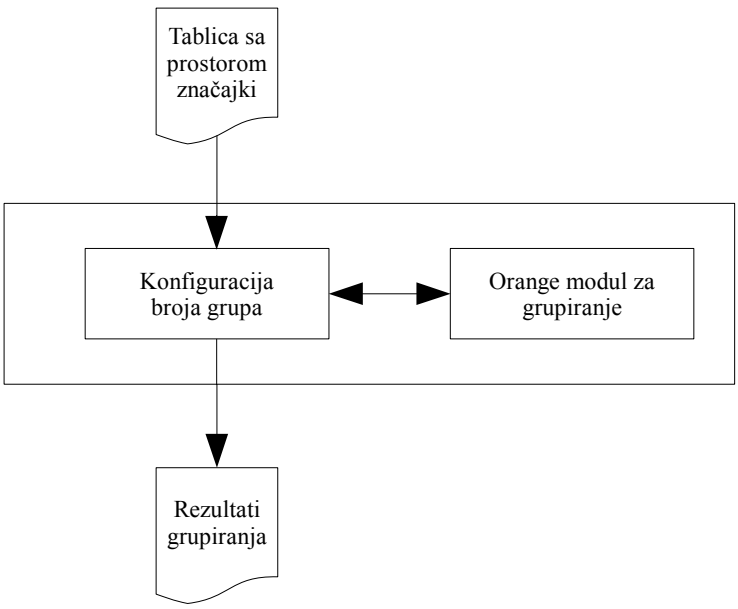
Ispis 4.5: Primjer dodatka za izvlačenje značajki HTML stranice

```
class ImgAttributeExtractor():
    """
    Extracts the number of img tags in a given string
    Returns Integer
    """

    def __init__(self):
        self.attributeName = "img_tags"
        self.attributeDescription = "number of img tags"
        self.attributeType = "discrete"

    def ExtractDataFromString(self,inputString):
        return inputString.count("<img ");
```

Tako implementiran okvir čini mogućim dodavanje novih značajki u prostor značajki bez potrebe za izmjenom kôda modula za izvlačenje značajki. Za ubacivanje novog dodatka ipak je potrebno poznavati programski jezik Python u kojem su dodaci napisani.

4.2.4. Grupiranje stranica

Modul za grupiranje stranica je ostvaren kao ovojnica prema već implementiranom modulu sustava Orange koji zadovoljava potrebe ovog sustava. Iz Orange sustava se koristi implementacija algoritma K srednjih vrijednosti (engl. K-means), a kroz impelementiranu ovojnicu se vrši kontrola parametara za algoritam. Ostavljena je mogućnost odabira drugih algoritama za grupiranje ili druge metode nad uzorcima. Arhitektura modula je prikazana na slici 4.6.



Slika 4.6: Arhitektura modula za grupiranje

Jedino ograničenje u Orange implementaciji odabranog algoritma koje je bilo potrebno zaobići je nemogućnost dinamičkog određivanja broja grupa. Zbog toga se kroz ovojnicu dinamički mijenja broj grupa u rasponu koje je definirao korisnik te se vrši grupiranje za svaki pojedini broj grupa. Rezultati se potom skupljaju i predstavljaju korisniku.

U sklopu rezultata grupiranja korisniku se pružaju slijedeće informacije:

- informacija o pripadnosti stranica pojedinim grupama
- informacije o sažetosti elemenata unutar grupa
- informacija o prosječnoj sažetosti grupa
- reprezentativni uzorci za svaku grupu

4.3. Datoteke

Prilikom dizajniranja sustava poseban je trud posvećen dizajniranju formata datoteka koje se u sustavu koriste. Strogo pridržavanje formatu datoteka nužno je za uspješan rad svih modula. Ostavljena je mogućnost jednostavne nadogradnje na trenutni format koji ne bi trebao rezultirati s nekompatibilnostima s trenutnim modulima. Unutar sustava se koriste četiri osnovne datoteke:

- datoteka s informacijama o aplikaciji
- konfiguracijska datoteka za fuzzer (nadogradnja na prethodnu datoteku)
- datoteka s prikupljenom HTML stranicom
- datoteka s tablicom značajki

U slijedećim potpoglavljima slijedi njihov detaljan opis.

4.3.1. Datoteka s informacijama o aplikaciji

Datoteka s informacijama o aplikaciji sadrži popis poveznica, parametara i korisnih informacija o aplikaciji koji su dobiveni korištenjem modula *crawler*. Informacije su opisane XML jezikom i ovdje je dan opis pojedinih elemenata. Ishodišni element cijelog XML dokumenta je *wsat* element unutar kojega je definiran prostor imena. Definicija prostora imena prikazana je u ispisu 4.6.

Ispis 4.6: Prostor imena

```
<wsat xmlns="http://www.zemris.fer.hr/wsat/0.9">
```

Svi ostali elementi su sadržani kao čvorovi ovog elementa. Ishodišni element trenutno sadrži pet mogućih elemenata prikazanih u ispisu 4.7, ali je moguća laka nadogradnja ukoliko se za to pokaže potreba. *Hostname* element sadrži ime poslužitelja na kojem se izvršava aplikacija. Element *separators* sadrži četiri parametra koji sadrže znakove koji se koriste za odvajanje pojedinih dijelova URL-a. Primjer potrebe za mijenjanje tih vrijednosti je objašenjen u ranijem poglavlju i prikazan u ispisu 3.1.

Ispis 4.7: Osnovna struktura XML dokumenta s opisom aplikacije

```
<?xml version="1.0" encoding="utf-8"?>
<wsat xmlns="http://www.zemris.fer.hr/wsat/0.9">
  <hostname>...</hostname>
  <separators param="&amp;" path=";" query="?" value="="/>
  <folders>
    ...
  </folders>
  <pages>
    ...
  </pages>
  <forms>
    ...
  </forms>
</wsat>
```

Element *folders* sadrži podelemente koji nabrajaju sve direktorije pronađene u poveznicama aplikacije. Primjer kako može izgledati cijeli *folders* element prikazan je u ispisu 4.8. Element je ovdje naveden zbog cjelovitosti opisa datoteke, ali se ne koristi u procesu detekcije potencijalnih ranjivosti.

Ispis 4.8: Primjer elementa folders u XML datoteci

```
<folders>
  <folder>http://testphp.acunetix.com/AJAX/</folder>
  <folder>http://testphp.acunetix.com/images/</folder>
  <folder>http://testphp.acunetix.com/secured/</folder>
</folders>
```

U *pages* elementu su pobrojane sve stranice unutar aplikacije koje su pronađene. Svaka stranica je opisana s jednim *page* elementom koji obavezno sadrži atribut *id* te podelement *url* u kojem je URL pronađene stranice. Ukoliko pronađena stranica posjeduje parametre, oni su pobrojani unutar *parameters* podelementa. Svaki parametar sadrži atribute koji opisuju ime parametra (*name*), tip parametra (*method*) te identifikator parametra (*id*). Postoje četiri metode ugrađivanja parametra u URL ili zahtjev stranice prema poslužitelju, a prikazane su u tablici 4.3.

Tablica 4.3: Metode ugrađivanja parametara

Metoda	Opis metoda ugrađivanja parametra
`get`	'klasični' parametri koji se šalju korištenjem GET metode
`post`	Parametri koji se šalju u tijelu zaglavlja prilikom POST metode
`path`	Parametri puta
`fragment`	Parametri oznake fragmenta na stranici

Parametri puta (engl. path parameters) se rijetko koriste, a najčešće služe za slanje sjedničkih identifikatora. Primjer parametra puta je prikazan u ispisu4.9.

Ispis 4.9: Primjer parametra puta

```
http://hostname.com/home;jsessionid=1111
```

Parametri oznake fragmenta na stranici se odvajaju znakom # i ne šalju se na poslužitelj, ali su uvršteni u sustav zbog mogućnosti ispitivanja DOM ubacivanja koje može koristiti i te parametre.

Unutar elementa *param*, nabrojane su i sve vrijednosti koje su pronađene uz taj parametar za tu određenu stranicu. Pronađene vrijednosti se navode unutar *value* elementa.

Važno je napomenuti da je sustav izgrađen na taj način da stranice razlikuje ne samo po imenu stranice već i po imenima parametara. Tako će npr. sustav stranice prikazane u ispisu 4.10. tretirati kao dvije različite stranice.

Ispis 4.10: Primjer dvije različite stranice za naš sustav

```
http://testphp.acunetix.com/artists.php?artist=3
http://testphp.acunetix.com/artists.php?artist=3&id=1
```

Primjer svih upravo opisanih elemenata i atributa je prikazan u ispisu 4.11.

Ispis 4.11: Primjer opisa stranice u XML dokumentu

```
<pages>
  ...
  <page id="9">
    <url>http://testphp.acunetix.com/search.php?test=query</url>
    <parameters>
      <param id="1" method="get" name="test">
        <values>
          <value>query</value>
        </values>
      </param>
    </parameters>
  </page>
  <page id="10">
    <url>http://testphp.acunetix.com/login.php</url>
  </page>
  ...
</pages>
```

Element *forms* je sličan prethodno opisanom elementu *pages*, ali sadrži malo više informacija o formama koje su pronađene i njihovim parametrima. Element se također sastoji od niza podelemenata od kojih svaki opisuje zasebnu formu, a u ovom kontekstu se element naziva *form*. Uz obavezan atribut identifikatora forme, svaki element sadrži četiri obavezna podelementa: *url*, *data*, *from* i *parameters*. Element *url* jednako kao i prije sadrži URL kojime se dohvaća stranica, a element *data* sadrži tijelo zaglavlja u kojem se nalaze imena i vrijednosti parametara koji se šalju POST metodom. Moguće je da forma sadrži i parametre koji se šalju kroz URL i parametre koji se šalju kroz tijelo zahtjeva.. U elementu *from* se nalazi URL stranice na kojoj je pronađena forma, a u elementu *parameters* se nalaze opisi svih parametara opisani na jednak način kao i kod elementa *page*. Dodatak opisima parametara je atribut *type* koji opisuje parametar na način kako je on opisan u HTML kôdu. Primjer opisa jedne forme prikazan je u ispisu 4.12.

Ukoliko se forma koristi kao forma za autentifikaciju korisnika na Web aplikaciju, podaci za autentifikaciju se također zapisuju u dokument. Elementu *form* se dodaje atribut *type* koji označava da je to forma za autentifikaciju, a potrebnim parametrima se dodaju vrijednosti

koje treba koristiti. Prikaz takve forme je dan u ispisu 4.13, a dijelovi koji označuju podatke za autentifikaciju su označeni masnim slovima.

Ispis 4.12: Primjer opisa forme u XML dokumentu

```
<forms>
  ...
  <form id="24">
    <url>http://testphp.acunetix.com/search.php?test=query</url>
    <data>searchFor=on&amp;goButton=go</data>
    <from>http://testphp.acunetix.com</from>
    <parameters>
      <param id="3" method="get" name="test">
        <values>
          <value>query</value>
        </values>
      </param>
      <param id="1" method="post" name="searchFor" type="text">
        <values>
          <value>on</value>
        </values>
      </param>
      <param id="2" method="post" name="goButton" type="submit">
        <values>
          <value>go</value>
        </values>
      </param>
    </parameters>
  </form>
</forms>
```

Ispis 4.13: Prikaz forme za autentifikaciju unutar XML dokumenta

```
<form id="31" type="login">
 <url>http://testphp.acunetix.com/userinfo.php</url>
 <data>uname=vanja&amp;NoName=&amp;pass=pass</data>
 <from>http://testphp.acunetix.com/login.php</from>
 <parameters>
   <param id="1" loginvalue="vanja" method="post" name="uname"
type="text">
      <values>
        <value>vanja</value>
      </values>
   </param>
   <param id="2" loginvalue="pass" method="post" name="pass"
type="password">
      <values>
        <value>pass</value>
      </values>
   </param>
   <param id="3" loginvalue="" method="post" name="NoName" type="submit">
      <values>
         <value>login</value>
      </values>
   </param>
 </parameters>
</form>
```

4.3.2. Konfiguracijska datoteka za fuzzer

Konfiguracijska datoteka za fuzzer je napravljena kao nadogradnja na postojeću datoteku s informacijama o aplikaciji. Elementi specifični za *fuzzer* se nalaze u posebnom prostoru imena s oznakom *fuzz*. Prva promjena u odnosu na osnovnu datoteku je definicija prostora imena unutar početnog elementa. Nova definicija je prikazan u ispisu 4.14.

Ispis 4.14: Nova definicija prostora imena

```
<wsat xmlns="http://www.zemris.fer.hr/wsat/0.9"
 xmlns:fuzz="http://www.zemris.fer.hr/wsat/fuzzer/0.5">
```

Promjene u dokumentu su ostvarene dodavanjem novih elemenata ili novih atributa u postojeće elemente. Unutar početnog elementa ubačena su dva nova važna elementa: *fuzz:settings* i *fuzz:values*. Prvi element opisuje postavke za modul fuzzer. Primjer takvog elementa i njegovih postavki prikazan je u ispisu 4.15, a u tablici 4.4 je dan opis svih trenutnih postavki i njihovih mogućih vrijednosti.

Ispis 4.15: Primjer elementa s postavkama za modul fuzzer

```
<fuzz:settings>
  ...
  <fuzz:setting name="defaultMethod">onebyone</fuzz:setting>
  <fuzz:setting name="defaultMaxRequestsPerParam">100</fuzz:setting>
  ...
</fuzz:settings>
```

Tablica 4.4: Popis svih postavki za modul fuzzer

Postavka	Opis
`defaultFuzzAll [True|False]`	Određuje da li se ispituju svi pronađeni parametri
`defaultFuzzSet <int>`	Određuje skup vrijednosti za ispitivanje koji se koristi ukoliko je uključeno ispitivanje svih parametara
`DefaultMethod [onebyone| pairwise|cross]`	Određuje tip spajanja parametara s vrijednostima za ispitivanje ukoliko to nije definirano unutar elementa sa stranicom ili formom koja se ispituje
`DefaultMaxRequestsPerParam <int>`	Određuje maksimalan broj ispitnih vrijednosti po parametru ukoliko to nije definirano unutar parametra za ispitivanje
`MaxRequestsPerPage <int>`	Određuje maksimalan broj ukupnog broja zahtjeva za pojedinu stranicu

Drugi važan element koji je ubačen u konfiguracijsku datoteku opisuje vrijednosti koje će se koristiti za ispitivanje parametara. Skup vrijednosti može biti opisan na dva načina. Prvi je način da se navede ime datoteke u kojoj se nalazi skup vrijednosti, a drugi je da se vrijednosti navedu unutar samog konfiguracijskog dokumenta. U oba slučaja skupovi moraju imati atribute koji označavaju identifikacijski broj skupa i ime skupa. Primjer takvih skupova dan je u ispisu 4.16.

Ispis 4.16: Definiranje skupova vrijednosti za ispitivanje.

```
<fuzz:values>
  <fuzz:set id="2" name="dictionary" file="dictionary.txt" />
  <fuzz:set id="3" name="numbers" file="numbers.txt" />
  <fuzz:set id="1" name="example">
    <fuzz:value>1</fuzz:value>
    <fuzz:value>2</fuzz:value>
    <fuzz:value>a</fuzz:value>
    <fuzz:value>b</fuzz:value>
    <fuzz:value>'</fuzz:value>
  </fuzz:set>
</fuzz:values>
```

Uz ova dva bitna elementa u konfiguracijsku datoteku se ubacuje i jedan element koji predstavlja primjer kako podesiti ispitivanje za određeni parametar. Taj element se ne mijenja, a prikazan je u ispisu 4.17.

Ispis 4.17: Element s primjerom konfiguracije testa za određeni parametar

```
<fuzz:example>
  <page fuzz:type="onebyone|pairwise|cross">
    <url>http://www.someapp.com/page.php?a=1&amp;b=2</url>
      <parameters>
        <param fuzz:max="yy" fuzz:set="x" method="get" name="a"/>
      </parameters>
  </page>
</fuzz:example>
```

U prethodnom ispisu se može vidjeti kako se odabrana stranica i parametri odabiru za ispitivanje. Unutar *page* ili *form* elementa potrebno je dodati novi atribut *fuzz:type* i odrediti način formiranja zahtjeva, tj, način kombiniranja parametara s njihovim pridruženim skupovima vrijednosti. Ukoliko se taj atribut izostavi, koristit će se tip definiran u postavkama na početku dokumenta. Za parametar koji je potrebno ispitati obavezno je pridijeliti skup vrijednosti za ispitivanje koji je definiran unutar *fuzz:values* elementa prikazanog u ispisu 4.16. Skup vrijednosti se veže s parametrom preko atributa oznake *fuzz:set* i vrijednosti koja označuje identifikacijski broj skupa. Opcionalno se može i odrediti nova vrijednost za maksimalni broj zahtjeva koji se može generirati, a to se postiže dodavanjem atributa *fuzz:max*. Primjer odabira parametra za ispitivanje je prikazan u ispisu 4.18. Masnim slovima su označeni atributi koji predstavljaju dio konfiguracije za testiranje.

Ispis 4.18: Primjer elementa za konfiguraciju ispitivanja

```
<page id="16" fuzz:type="onebyone">
  <url>http://testphp.acunetix.com/artists.php?artist=1</url>
  <parameters>
    <param id="1" method="get" name="artist" fuzz:set="3" fuzz:max="50">
      <values>
        <value>1</value>
        <value>3</value>
        <value>2</value>
      </values>
    </param>
  </parameters>
</page>
```

4.3.3. Datoteke s prikupljenim stranicama

Datoteke s prikupljenim stranicama imaju posebno definiran format zapisa. Format je prikazan u ispisu 4.19 i sastoji se od dva dijela. Prvi dio su komentari našeg sustava koji sadrže informacije kako je stranica dohvaćena, a drugi dio je HTML kôd stranice. Komentari moraju od HTML kôda biti odvojeni jednom praznom linijom.

Ispis 4.19: Format prikupljene stranice

```
<!--[wsat_comment]-->
<!--[wsat_comment]-->
<!--[wsat_comment]-->

<html>
 <head>
  <title>Naslov</title>
...
```

Komentari imaju također definiran format a sastoje se od imena stavke i vrijednosti odvojenih znakom dvotočka. Trenutno se koriste samo dva komentara koji određuju URL na kojem je dohvaćena stranica i vrijednost podatkovnog dijela zahtjeva ukoliko je on dohvaćan POST metodom. Primjer komentara je prikazan u ispisu

Ispis 4.20: Primjer komentara u prikupljenim stranicama.

```
<!--url:http://testphp.acunetix.com/search.php?test=query-->
<!--data:searchFor=on&goButton=go-->
```

Ukoliko se pokaže potreba za dodavanjem novih informacija o dohvaćenoj stranici, mogu se samo dodati novi komentari. Jedan od takvih primjera je dodavanje specifičnih zaglavlja koji su korišteni prilikom dohvaćanja stranice. Cilj ovih komentara je opisati kako je stranica dohvaćena i omogućiti ponovno izvršavanje tog dijela testa. Ukoliko je pronađena potencijalna ranjivost, na temelju komentara se može odrediti ranjivi parametar.

4.3.4. Značajke stranica

Za proces razvrstavanja stranica u grupe prema njihovoj sličnosti odlučeno je koristiti već gotovu implementaciju algoritma koja dolazi sa sustavom Orange [35]. Orange već specificira kako treba izgledati ulazna datoteka s vektorima koji opisuju podatke, pa je ta specifikacija preuzeta i u ovom sustavu. Na taj način se vrlo jednostavno može zamijeniti algoritam grupiranja uzoraka bez da se mijenja ulazna datoteka. Specifikacije ulazne datoteke je generička i jednaka za sve algoritme. Primjer ulazne datoteke prikazan je u ispisu 4.21.

Ispis 4.21: Primjer ulazne datoteke u sustav Orange

```
age             prescription    astigmatic    tear_rate      lenses
discrete        discrete        discrete      discrete       discrete
                                                             class
young           myope           no            reduced        none
young           myope           no            normal         soft
young           myope           yes           reduced        none
young           myope           yes           normale        hard
young           hypermetrope    no            reduced        none
```

U prvom retku su prikazana imena atributa. Imena mogu sadržavati sve znakove (uključujući i razmak) osim znakova prelaska u novi red (CR, LF), znaka NULL i tabulatora. Imena se odvajaju tabulatorima.

Drugi redak sadrži tip atributa. Popis svih tipova atributa i njihovih pripadajućih ključnih riječi je prikazan u ispisu 4.22. Alternativno, umjesto tipa *discrete*, moguće je nabrojati sve vrijednosti koje se smiju pojaviti. Vrijednosti se odvajaju razmacima, dok se tipovi odvajaju tabulatorima. Trenutni algoritmi implementirani u sustavu Orange koriste samo atribute diskretnog i kontinuiranog tipa, a tekstualni tip se može koristiti kao dodatni atribut za opis primjera ili se može koristiti u drugim implementacijama algoritama.

Ispis 4.22: Popis mogućih vrijednosti za opis atributa

Tip vrijednosti	**Dugačka oznaka tipa**	**Kratka oznaka tipa**
Diskretne vrijednosti atributa	`discrete`	`d`
Kontinuirane vrijednosti atributa	`continuous`	`c`
Tekstualne vrijednosti	`string`	

Treći redak dodatno opisuje atribute. Tako u našem primjeru zadnji stupac sadrži riječ *class* koja označava da taj stupac označava klasu u koju treba grupirati uzorke. Popis mogućih vrijednosti je prikazan u ispisu 4.23. Sve vrijednosti se odvajaju tabulatorom.

Ispis 4.23: Popis mogućih vrijednosti za dodatna objašnjenja atributa

Dodatno objašnjenje	**Dugačka oznaka**	**Kratka oznaka**
klasa	`class`	`c`
atribut treba zanemariti	`ignore`	`i`
atribut je opis uzorka (ne koristi se u algoritmima klasifikacije ili grupiranja)	`meta`	`m`

Naš sustav ne koristi uzorke za učenje i ne posjeduje a-priori znanje kojoj bi klasi koji uzorci trebali pripadati pa tako u našim datotekama ne postoji stupac koji određuje klasu. Za vrijednosti atributa koristimo brojčane oznake, a primjer jedne datoteke spremne za grupiranje stranica je prikazan u ispisu 4.24. Svaki stupac tablice prikazuje rezultate jednog dodatka modula za izvlačenje stranica.

Ispis 4.24: Primjer pune tablice

file_name string meta	href d	input d	td d	simple d	div d	img d	size d	P d	word d
page001.html	25	0	39	5	1	49	19688	4	404
page002.html	23	0	42	14	1	50	22484	4	534
page003.html	23	0	45	10	1	59	21740	4	454
page004.html	24	0	42	8	1	50	23584	4	593
page005.html	27	0	49	8	1	71	22374	4	430
page006.html	21	0	43	7	1	53	20812	4	437
page007.html	35	0	33	12	1	39	21090	5	450
page008.html	38	0	41	38	1	47	22327	4	474
page009.html	21	0	42	10	1	46	20250	4	436
page010.html	19	0	39	13	1	40	19666	4	438
page011.html	18	0	39	4	1	40	20056	4	479
page012.html	33	0	39	6	1	40	21021	4	476
page013.html	25	0	45	15	1	55	24747	4	668
page014.html	20	0	41	4	1	44	20163	4	465
page015.html	20	0	41	7	1	44	19770	4	412
page016.html	20	0	41	4	1	44	19951	4	448
page017.html	16	0	38	4	1	36	17485	4	375
page018.html	16	0	38	4	1	36	17463	4	375
page019.html	17	0	38	5	1	38	17656	4	378
page020.html	18	0	38	4	1	36	17480	4	375
page021.html	0	0	0	0	0	0	117	0	5
page022.html	0	0	0	0	0	0	117	0	5
page023.html	0	0	0	0	0	0	116	0	5

4.4. Korištenje sustava

Sustav se koristi iz komandne linije i trenutno ne postoji implementirano grafičko sučelje. Moguće je koristiti module i iz Python interpretira ili pisanjem Python skripti, ali to zahtjeva poznavanje programskog jezika Python i internog rada modula pa ovdje neće biti objašnjeno.

Korištenje sustava se može podijeliti u dva koraka s jednim međukorakom u kojem se definiraju postavke za dohvaćanje stranica. Napredniji korisnici mogu koristiti svaki modul posebno te dodatno podešavati opcije za svaku fazu obrade. Prvi korak je korištenje modula *crawler* za dohvaćanje informacija o zadanoj Web aplikaciji. Modul crawler je potpuno neovisan o ostalim modulima, a prijenos informacija u slijedeću fazu obrade se odvija putem XML međudatoteke. U ispisu 4.25. prikazana je sintaksa poziva modula.

Ispis 4.25: Sintaksa poziva modula crawler

```
$ python crawler.py [options] http://www.posluzitelj.com
```

Ime datoteke koja pokreće modul je *crawler.py*, parametar koji prima je adresa Web aplikacije, a opcije su prikazane u tablici 4.5.

Tablica 4.5: Popis opcija za modul crawler

Opcija	Duga opcija	Opis
`-h`	`--help`	Ispisuje poruku o korištenju modula
`-s <url>`	`--start <url>`	Definicija početnog URL-a
`-x <url>`	`--exclude <url>`	Specifikacija zabranjenog URL-a, dozvoljeno korištenje modifikatora *
`-c <file>`	`--cookie <file>`	Korištenje kolačića iz zadane datoteke
`-l [guess|all]`	`--login [guess|all]`	Podešava modul da traži od korisnika podatke za prijavu na sustav kada se nađe forma. Za *guess* se koristi heuristika za odabir forme, a kod *all* se korisniku nude sve forme na odabir
`-d <int>`	`--depth <int>`	Definiranje dubine pretraživanja
`-v <int>`	`--verbose <int>`	Definiranje razine ispisa <0,2>
`-t <int>`	`--timeout <int>`	Definiranje maksimalnog vremena čekanja na zahtjev

Moguće je adresu Web aplikacije pisati skraćeno, bez oznake protokola, i u tom se slučaju pretpostavlja korištenje HTTP protokola. Za aplikacije na kojima postoji autentifikacijski proces, moguće je unijeti podatke za autentifikaciju na dva načina: putem kolačića i putem interaktivnog mehanizma za autentifikaciju. Ako korisnik odluči koristiti kolačiće, to je moguće navodeći datoteku u kojoj su oni spremljeni. Primjer takvog korištenja modula prikazan je u ispisu 4.26. U tom slučaju se očekuje da su podaci za autentifikaciju ispravni.

Ispis 4.26: Autentifikacija pomoću kolačića

```
$ python crawler.py -c cookies.lwp testphp.acunetix.com
```

Ukoliko korisnik ne posjeduje kolačić za autentifikaciju, moguće je koristiti interaktivni mehanizam za autentifikaciju. Mehanizam se aktivira --*login* opcijom i on tada prati informacije koje se pronalaze analizom dohvaćenih stranica i za sve pronađene forme ispisuje njihove podatke i traži od korisnika potvrdu da se radi o formi za autentifikaciju. Kada korisnik odgovori potvrdno, od njega se traži da unese vrijednosti za sve parametre forme i zatim se ponovno pokreće proces analize stranica. U slučaju negativnog odgovora, nastavlja se s potragom za ispravnom formom.

Mehanizam se služi i s jednostavnom heurističkom metodom za pretpostavke da li forma služi za autentifikaciju. Metoda se sastoji od provjeravanja tipova parametara koji su skupljeni iz HTML kôda, te ako se pronađe parametar tipa *password*, mehanizam zaključuje da bi to mogla biti forma za autentifikaciju. Za opciju --*login* su moguće dvije vrijednosti. Kada se koristi vrijednost *guess*, mehanizam se služi isključivo heurističkom metodom i korisniku prikazuje samo forme za koje pretpostavi da služe za autentifikaciju. Korištenjem vrijednosti *all*, mehanizam korisniku nudi na odabir sve forme koja pronađe. Primjer pozivanja modula uz korištenje interaktivne autentifikacije je dan u ispisu 4.27.

Ispis 4.27: Interaktivna autentifikacija

```
$ python crawler.py -l guess testphp.acunetix.com
```

Rezultat analize aplikacije je XML dokument s popisom pronađenih direktorija, poveznica, formi i informacija o njima. Konfiguracijski dokument za modul *fuzzer* je temeljen na ovom

dokumentu, ali sadrži i dodatne elemente. Da bi se olakšalo njegovo kreiranje, moguće je dobiveni dokument provesti kroz međukorak u kojem se on priprema za konfiguraciju *fuzzer* modula. To se izvodi naredbom prikazanom u ispisu 4.28.

Ispis 4.28: Pripremanje konfiguracijskog dokumenta

```
$ python fuzzer.py -i appinfo.xml -o config.xml
```

Opcijom *-i* se definira dokument koji smo dobili od modula *crawler*, iz njega se vade relevantne informacije i u dokument se dodaju elementi potrebni za konfiguraciju. Pripremljeni konfiguracijski dokument se sprema pod imenom navedenom uz *-o* opciju. Moguće je koristiti dodatnu opciju koja učitava skup vrijednosti i uključuje ga izravno u konfiguracijski dokument. U tom slučaju naredba izgleda kao u ispisu 4.29. Datoteka iz koje se učitava skup vrijednosti za ispitivanje treba sadržavati po jednu ispitnu vrijednost u svakom redu ili vrijednosti moraju biti odvojene redovima s graničnikom <--*wsat*-->. Modul sam zaključuje o kojem se formatu datoteke radi.

Ispis 4.29: Pripremanje konfiguracijskog dokumenta

```
$ python fuzzer.py -i appinfo.xml -s numbers.txt -o config.xml
```

Nakon pripreme dokumenta, od korisnika se još očekuje da odredi da li želi provoditi analizu nad svim parametrima ili želi odabrati samo određene te podesiti pripadne vrijednosti koje će se koristiti uz parametre. Podešavanje parametara je objašnjeno u poglavlju s opisom konfiguracijske datoteke.

Jednom kada je konfiguracija dovršena, podaci su spremni za drugi korak. U drugom koraku se izvode ostale tri faze rada odjednom. Moduli i dalje komuniciraju preko međudokumenata, ali je njihova interakcija izvedena interno kako bi se smanjila uloga korisnika koja nije u toj mjeri potrebna u ovim fazama. Zasebno pokretanje završnih faza je također moguće i omogućuje zahtjevnijim korisnicima dodatne konfiguracijske mogućnosti. Drugi korak rada se pokreće također kroz modul *fuzzer*, ali se koriste drugi parametri. Opća sintaksa pokretanja modula je prikazana u ispisu 4.30, a sve opcije su navedene u tablici 4.6.

Ispis 4.30: Sintaksa poziva modula fuzzer

```
$ python fuzzer.py [options]
```

Tablica 4.6: Popis opcija za modul fuzzer

Opcija	Duga opcija	Opis
-h	--help	Ispisuje poruku o korištenju modula
-f	--fuzz	Pokretanje akcije dohvaćanja stranica
-c <file>	--cookie <file>	Korištenje kolačića iz zadane datoteke
-d <path>	--dir <path>	Definiranje početnog direktorija za spremanje rezultata
-i <file>	--info <file>	Konfiguracijska datoteka za fuzzer
-o <file>	--output <file>	Ispis konfiguracijske datoteke
-s <file>	--set <file>	Učitava skup vrijednosti za ispitivanje
-v <int>	--verbose <int>	Definira razine ispisa informacija o radu modula

Primjer pokretanja drugog koraka rada sustava je naveden u ispisu 4.31. Pomoću *-i* opcije se učitava i parsira konfiguracijski dokument i pripremaju se svi zahtjevi za dohvaćanje

stranica. Ukoliko je opcijom *-d* naveden početni direktorij za spremanje rezultata, hijerarhija direktorija će se formirati unutar njega, u protivnom će se rezultati spremati u trenutni direktorij iz kojeg je pokrenut modul. Najbitnija opcija je *-f* koja pokreće proces dohvaćanja stranica, izvlačenja značajki i grupiranja.

Ispis 4.31: Pokretanje drugog dijela procesa

```
$ python fuzzer.py -i config.xml -f -d results
```

Unutar direktorija u koji su spremljene dohvaćene stranice kreiraju se dodatne dvije datoteke. Prva je *features.tab* u kojoj se nalazi tablica sa značajkama i predstavlja međudokument između modula za izvlačenje značajki i modula za grupiranje. Rezultati grupiranja se upisuju u datoteku imena *groups.txt* u kojoj se nalaze sve relevantne informacije o grupiranju dokumenata. Pregledom tog dokumenta korisnik može odrediti kritične točke Web aplikacije i usmjeriti daljnje istraživanje na pojedine segmente.

Iako je upravljanje s dvije završne faze obrade izvedeno automatizirano, korisniku je ostavljena opcija da module koristi zasebno. Sintaksa korištenja modula za izvlačenje značajki je prikazana u ispisu 4.32.

Ispis 4.32: Sintaksa poziva modula za izvlačenje značajki

```
$ python featureextract.py [options] <dir>
```

Prvi argument iza naziva modula je put do direktorija u kojem se nalaze prikupljene stranice. U taj direktorij će se spremiti i rezultati modula.

Postoji tek jedna opcija koja ispisuje dodatne informacije o radu modula. Poziv je prikazan u tablici 4.7.

Tablica 4.7: Opcije modula za izvlačenje značajki

Opcija	Duga opcija	Opis
`-h`	`--help`	Ispisuje poruku o korištenju modula
`-v`	`--verbose`	Uključuje ispis informacija o radu modula

Grupiranje uzoraka moguće je pokrenuti odvojeno, korištenjem modula za grupiranje. Općenita sintaksa pokretanja modula prikazana je u ispisu 4.33. Prvi argument modulu je ime datoteke s tablicom značajki. Opcije su prikazane u tablici 4.8.

Ispis 4.33: Sintaksa poziva modula za grupiranje

```
$ python clusterer.py [options] <features_file>
```

Tablica 4.8: Popis opcija za modul za grupiranje

Opcija	Duga opcija	Opis
`-h`	`--help`	Ispisuje poruku o korištenju modula
`-f <int>`	`--from <int>`	Određuje minimalan broj grupa
`-t <int>`	`--to <int>`	Određuje maksimalan broj grupa
`-o <file>`	`--output <file>`	Određuje datoteku za ispis rezultata
`-v`	`--verbose`	Ispisuje rezultate na ekran

Rezultat procesa je datoteka s informacijama o grupiranju stranica. Na samom početku su nabrojane stranice koje su se ispitivale. Uz svaku stranicu je navedeno ime datoteke u kojoj je stranica pohranjena, zatim vektor koji predstavlja niz značajki stranice te dodatni komentari o stranici koji su spremljeni uz sam HTML kôd stranice. U drugom dijelu su prikazane informacije o grupiranje za odabrane brojeve grupa. Primjer rezultata za grupiranje stranica u četiri grupe prikazan je u ispisu 4.34.

Ispis 4.34: Informacije o grupiranju

```
*** 4 ***
mapping [1, 1, 1, 2, 3, 3, 3, 4, 4, 4]
tightness per groups: ['0.97814', '0.00000', '0.99437', '1.00000']
avg tightness: 0.89175
group 1:  center: 002.html    all: ['001.html', '002.html', '003.html']
group 2:  center: 004.html    all: ['004.html']
group 3:  center: 006.html    all: ['005.html', '006.html', '007.html']
group 4:  center: 010.html    all: ['008.html', '009.html', '010.html']
```

Nakon naslova u kojem se navodi broj grupa, uz tekst *mapping* navode se grupe kojima su pridružene pojedine stranice. Zatim se navode informacije o sažetosti elemenata unutar grupe i srednjoj sažetosti grupa. Grupe sa samo jednim dodijeljenim elementom će imati sažetost jednaku nuli, a grupe koje sadrže identične stranice (tj. identične vektore značajki) će imati sažetost jednaku jedan. Na kraju su pobrojane sve grupe te je za svaku grupu naveden predstavnik koji se nalazi najbliže središtu grupe te popis svih stranica pridruženih pojedinoj grupi. Predstavnici grupa su označeni tekstom *center* i služe za vizualni pregled karakteristične stranice iz navedene grupe.

Za pomoć pri kreiranju skupova vrijednosti za ispitivanje implementiran je dodatni modul koji ih generira. Opća sintaksa pozivanja modula je prikazana u ispisu

Ispis 4.35: Sintaksa poziva modula za generiranje ispitnih skupova

```
$ python generator.py <modes> [options]
```

Sintaksa se sastoji od opcija koje određuju način generiranja vrijednosti i dodatnih opcija. Uz svaki način rada je potrebno navesti argumente odvojene razmacima. Popis načina i njihovih parametara te kratko objašnjenje prikazano je u tablici 4.9.

Tablica 4.9: Načini generiranja elemenata

Duga opcija	**Argumenti**	**Opis**
`--seqnum`	`<start>,<end>,<step>`	Slijedni brojevi s definiranim korakom
`--randnum`	`<start>,<end>,<count>`	Nasumični brojevi
`--charblocks`	`<char>,<min>,<max>, <step>`	Blokovi znakova koji se ponavljaju
`--bruteforce`	`<charset>,<min>,<max>`	Sve kombinacije znakova
`--randfilechunks`	`<filename>,<min>,<max>, <count>`	Nasumični dijelovi datoteke definirane duljine
`--filelines`	`<filename>,<count>`	Redci datoteke
`--xssdb`		Baza XSSDB ranjivosti

Posljednji način generiranja vrijednosti za ispitivanje zapravo ne generira vrijednosti već dohvaća s Interneta XSSDB bazu ranjivosti [44] te ju parsira i generira oblik prigodan za korištenje u našem sustavu.

Opcije koje se nude uz modul su prikazane u tablici 4.10.

Tablica 4.10: Opcije modula za generiranje ispitnih skupova

Opcija	**Duga opcija**	**Opis**
`-h`	`--help`	Ispisuje poruku o korištenju modula
`-v`	`--verbose`	Ispisuje skup vrijednosti na ekran
`-s <int>`	`--sample <int>`	Ispisuje nasumični uzorak vrijednosti iz skupa
`-o <file>`	`--output <file>`	Sprema skup vrijednosti u datoteku.

5. Eksperimentalni rezultati

Eksperimenti sa sustavom su izvedeni na testnim stranicama tvrtke Acunetix [45]. Tvrtka se bavi sa sigurnošću Web aplikacija i ima na tržištu vlastitu aplikaciju za traženje ranjivosti u Web aplikacijama. Aplikacija je komercijalna pa je nismo proučavali u sklopu rada, ali su zato iskorištene njihove testne stranice. Na raspolaganju su tri aplikacije pisane u različitim tehnologijama, a aplikacije su napisane tako da posjeduju razne sigurnosne propuste. Mogu se pronaći na Web adresama prikazanim u ispisu 5.1.

Ispis 5.1: Web aplikacije za testiranje

```
http://testphp.acunetix.com
http://testasp.acunetix.com
http://testaspnet.acunetix.com
```

U prvom dijelu poglavlja će biti opisani problemi s kojima smo se susreli tijekom implementacije i ispitivanja sustava, a zatim će biti dani rezultati eksperimenata s gotovim sustavom. Pitanja na koja je trebalo odgovoriti su:

- Kako odrediti značajke koje dobro opisuju Web stranice?
- Koji skup vrijednosti koristiti za ispitivanje?
- Kako razlikovati vizualno slične stranice?

5.1. Određivanje značajki Web stranica

Problem određivanja karakterističnih značajki koje dovoljno dobro opisuju stranicu svojstven je svim metodama raspoznavanja uzoraka. Uzorci su u našem slučaju HTML kôdovi dohvaćenih Web stranica i postavlja se pitanje koje su to značajke koje se trebaju koristiti za grupiranje uzoraka, a koje će omogučiti detekciju ranjivosti. Prilikom ispitivanja sustava odabrane su slijedeće značajke stranica:

- broj poveznica
- broj različitih poveznica
- broj pojavljivanja *div* elementa
- broj pojavljivanja *table* elementa
- broj pojavljivanja *td* elementa
- broj pojavljivanja *img* elementa
- broj pojavljivanja *input* elementa
- broj pojavljivanja *p* elementa
- broj pojavljivanja jednostavnih elemenata koji ne sadrže nikakav atribut
- naznake korištenja puta u tekstu (riječi odvojene kosim crtama)
- broj riječi na stranici
- dužina stranice

Pokazano je da dodavanje novih značajki samo pridonosi kvaliteti grupiranja stranica. S ovakvim skupom značajki provedeno je nekoliko eksperimenata. U prvom su se eksperimentu promatrale ručno prikupljene stranice s tri različite Web aplikacije (ispis 5.2) i promatralo se za koji broj grupa su grupe najviše sažete. Vizualnom provjerom su zatim provjereni rezultati grupiranja.

Ispis 5.2: Aplikacije korištene za prvi eksperiment

```
http://www.aozeljeznicar.hr
http://www.neoinfo.hr
http://testphp.acunetix.com
```

Algoritam je uspješno razvrstao stranice i za 3 grupe dao najveću prosječnu sažetost grupa.

Ispis 5.3: Najbolji rezultat grupiranja

```
*** 3 ***
mapping [1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3]
tightness per groups: ['0.84952', '0.55363', '0.88794']
avg tightness: 0.73584
group 1:  center: aoz.4.html    all: ['aoz.1.html', 'aoz.2.html', 'aoz.
   3.html', 'aoz.4.html']
group 2:  center: neoinfo.4.html    all: ['neoinfo.1.html', 'neoinfo.
   2.html', 'neoinfo.3.html', 'neoinfo.4.html', 'neoinfo.5.html']
group 3:  center: testphp.2.html    all: ['testphp.1.html', 'testphp.
   2.html', 'testphp.3.html']
```

Vidimo da druga grupa uzoraka ima raspršenije elemente od ostale dvije grupe, ali su oni i dalje ispravno grupirani u svoju grupu. Za manji broj grupa, algoritam daje lošiji rezultat prikazan u ispisu 5.4.

Ispis 5.4: Rezultat grupiranja za manji broj grupa

```
*** 2 ***
mapping [1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2]
tightness per groups: ['0.88883', '0.45041']
avg tightness: 0.59655
group 1:  center: aoz.4.html    all: ['aoz.1.html', 'aoz.2.html', 'aoz.
   3.html', 'aoz.4.html']
group 2:  center: neoinfo.3.html    all: ['neoinfo.1.html', 'neoinfo.
   2.html', 'neoinfo.3.html', 'neoinfo.4.html', 'neoinfo.5.html',
   'testphp.1.html', 'testphp.2.html', 'testphp.3.html']
```

Web stranice jedne od aplikacija su grupirane u prvu grupu i sažetost te grupe je visoka, no preostale stranice dviju različitih aplikacija su grupirane zajedno i to rezultira niskom sažetošću grupe, odnosno, velikom raspršenošću uzoraka. Srednja sažetost grupa je manja od prethodnog slučaja.

Ukoliko povećamo broj grupa u koje algoritam treba dodijeliti stranice, dobit će se slijedeći rezultati.

Ispis 5.5: Rezultat grupiranja za prevelik broj grupa

```
*** 4 ***
mapping [1, 1, 1, 1, 2, 2, 2, 2, 3, 4, 4, 4]
tightness per groups: ['0.83359', '0.48104', '0.00000', '0.81696']
avg tightness: 0.64245
group 1:  center: aoz.4.html    all: ['aoz.1.html', 'aoz.2.html', 'aoz.
   3.html', 'aoz.4.html']
group 2:  center: neoinfo.4.html    all: ['neoinfo.1.html', 'neoinfo.
   2.html', 'neoinfo.3.html', 'neoinfo.4.html']
group 3:  center: neoinfo.5.html    all: ['neoinfo.5.html']
group 4:  center: testphp.2.html    all: ['testphp.1.html', 'testphp.
   2.html', 'testphp.3.html']

*** 5 ***
mapping [1, 1, 1, 1, 2, 2, 3, 3, 4, 5, 5, 5]
tightness per groups: ['0.80486', '0.46889', '0.45185', '0.00000',
   '0.81696']
avg tightness: 0.62598
group 1:  center: aoz.4.html    all: ['aoz.1.html', 'aoz.2.html', 'aoz.
   3.html', 'aoz.4.html']
group 2:  center: neoinfo.2.html    all: ['neoinfo.1.html', 'neoinfo.
   2.html']
group 3:  center: neoinfo.4.html    all: ['neoinfo.3.html', 'neoinfo.
   4.html']
group 4:  center: neoinfo.5.html    all: ['neoinfo.5.html']
group 5:  center: testphp.2.html    all: ['testphp.1.html', 'testphp.
   2.html', 'testphp.3.html']
```

Vidimo da je srednja vrijednost sažetosti grupa u oba slučaja manja od one početne prikazane u ispisu 5.3.

Kada je eksperiment ponovljen s većim brojem stranca iz svake aplikacije, najbolji rezultat je i dalje bio upravo za tri grupe uzoraka. Isječak iz rezultata prikazan je u ispisu 5.6. Ovime je pokazano da sustav može pravilno razvrstavati stranice koje su međusobno dosta različite po svojoj strukturi.

Ispis 5.6: Rezultati grupiranja s većim brojem stranica

```
*** 2 ***
mapping [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 2, 2, 2, 1, 1, 2,
   2, 2, 2, 2, 2, 2, 2, 2, 2]
tightness per groups: ['0.49930', '0.68854']
avg tightness: 0.59392
...

*** 3 ***
mapping [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3,
   3, 3, 3, 3, 3, 3, 3, 3, 3]
tightness per groups: ['0.71715', '0.66986', '0.92532']
avg tightness: 0.77078
...

*** 4 ***
mapping [1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4,
   4, 4, 4, 4, 4, 4, 4, 4, 4]
tightness per groups: ['0.61170', '0.45470', '0.64825', '0.92532']
avg tightness: 0.71272
...

*** 5 ***
mapping [1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 3, 4, 3, 4, 3, 4, 4, 4, 3, 3, 5,
   5, 5, 5, 5, 5, 5, 5, 5, 5]
tightness per groups: ['0.61170', '0.45470', '0.60667', '0.53914',
   '0.90715']
avg tightness: 0.68155
...
```

Usporedba grupiranja s manjim i većim skupom vrijednosti za ispitivanje prikazana je u tablici 5.1. Masnim slovima su označene najveće vrijednosti.

Tablica 5.1: Usporedba grupiranja za različite veličine skupova

Broj grupa	Srednja sažetost – mali skup (12)	Srednja sažetost – veliki skup (30)
2	0.59655	0.59392
3	**0.73584**	**0.77078**
4	0.64245	0.71272
5	0.62598	0.68155

U drugom će eksperimentu biti pokazano kako sustav razvrstava stranice jedne Web aplikacije koje su međusobno puno sličnije. Za ispitivanje će se koristiti stranica čiji je URL prikazan u ispisu 5.7 s različitim vrijednostima za parametar *cat*.

Ispis 5.7: URL stranice za ispitivanje u drugom eksperimentu

```
http://testphp.acunetix.com/listproducts.php?cat=1
```

Pregledom mogućih izgleda stranice za različite vrijednosti parametra *cat* ustanovljeno je da postoje četiri osnovne grupe u koje bi korisnik razvrstao stranice:

- za ispravan identifikator kategorije prikazuju se stranice s proizvodima. One se razliku po količini proizvoda i količini teksta uz njih

- za neispravan identifikator se prikazuje prazna stranica sa zaglavljem i izbornicima Web aplikacije
- ukoliko se kao identifikator pokuša unijeti slovo, prikazat će se poruka o grešci od SQL poslužitelja
- za unos nekog znaka koji nije ni slovo ni brojka, prikazuje se greška koja je različita od prethodne

Eksperiment se provodio nekoliko puta a u slijedećem ispisu su prikazani rezultati grupiranja za broj grupa između 2 i 7 za dva različita skupa vrijednosti. Prvi skup se sastojao od približno 15 vrijednosti dok je drugi skup imao približno 45 različitih vrijednosti. Masnim slovima su označene najveće vrijednosti.

Tablica 5.2: Prikaz rezultata

Broj grupa	Srednja sažetost – mali skup (13)	Srednja sažetost – veliki skup (44)
2	**0.85876**	0.56569
3	0.65678	0.79375
4	0.67424	0.88113
5	0.83519	**0.92359**
6	0.58716	0.81500
7	0.51282	0.83338

Ponovnim vizualnim pregledom stranica je zaključeno da u prvoj skupini u kojoj se nalaze stranice s prikazom proizvoda postoje stranice koje se bitno razlikuju po količini proizvoda u odabranoj kategoriji. Zbog toga sustav ima problema sa svrstavanjem takvih stranica u istu grupu i razdvaja ih u dvije. Za očekivati je da sustav stranice koje bi korisnik možda i dodijelio u istu grupu odvoji u dvije ili više grupa ovisno o značajkama stranica. Iz tablice 5.2 se još vidi da rezultati dosta variraju u odnosu na veličinu skupa za ispitivanje što je problem koji će biti analiziran u slijedećem poglavlju. Za većinu grupa je srednja sažetost porasla s rastom uzoraka, što znači da su grupe kompaktnije i da su uzorci bolje razvrstani. Jedini pad sažetosti se pokazuje u odabiru samo dvije grupe za grupiranje što ukazuje da je broj grupa premali za kvalitetno grupiranje pa se uzorci unutar grupa uvelike razlikuju i smanjuju sažetost grupe. Navedeni zaključci se mogu vidjeti i u prvom eksperimentu čija je usporedba prikazana u tablici 5.1.

Pokazano je da odabrane značajke dovoljno dobro opisuju izgled stranice i da se mogu nositi s grupiranjem stranica različitih aplikacija i grupiranjem stranica jedne aplikacije.

5.2. Skup vrijednosti za ispitivanje

U prethodnom je poglavlju pokazano koliko je bitan odabir skupa vrijednosti za ispitivanje i koliko utječe na rezultate. Pokazano je da se s rastom skupa poboljšavaju rezultati grupiranja, ali nije dan odgovor na pitanje kakve su vrijednosti bile odabrane u skupu. Za prethodne primjere postojala su četiri osnovna tipa vrijednosti za ispitivanje:

- brojevi koji su vjerojatno ispravni identifikatori kategorija proizvoda
- brojevi za koje je malo vjerojatno da su ispravni identifikatori kategorija proizvoda

- vrijednosti sa slovima za koje se očekuje poruka o grešci
- vrijednosti s ostalim znakovima za koje se očekuje alternativna poruka o greški

Sve četiri skupine su bile jednako zastupljene pa je bilo i za očekivati da različiti tipovi prikupljenih stranica budu jednako zastupljeni. U ovom poglavlju će biti prikazani eksperimenti s raznim udjelima pojedinih tipova vrijednosti u skupu za ispitivanje. Zbog jednostavnosti će biti opisana analiza nad istom stranicom kao u prethodnom primjeru, prikazanom u ispisu 5.7.

Prvo smo za svaki tip vrijednosti definirali skup iz kojeg će se uzimati vrijednosti za ispitivanje:

$$s_1 \subset (potencijalno\ neispravni\ identifikatori)$$
$$s_2 \subset (kombinacija\ slova\ približno\ jednake\ duljine)$$
$$s_3 \subset (kombinacija\ znakova\ približno\ jednake\ duljine)$$

U prvom dijelu smo promatrali kako se ponaša sustav kada je u skupu vrijednosti za ispitivanje izrazito prisutan jedan od navedenih podskupova. Sažetost skupova po veličini grupa prikazana je u tablici 5.3.

Tablica 5.3: Grupiranje sa zastupljenim podskupovima

Broj grupa	3	4	5	6	7
Skup S_1	0.88776	**0.89710**	0.86041	0.86207	0.06897
Skup S_2	0.89494	**0.92373**	0.92041	0.89474	0.05263
Skup S_3	0.91807	**0.92092**	0.91689	0.92041	0.89474

Iako na prvi pogled izgleda da su stranice ispravno grupirane u četiri grupe, pregledom elemenata grupe ustanovljeno je da je sustav u sva tri slučaja razdijelio dvije različite stranice u svoje grupe, u trećoj grupi su se nalazile stranice dohvaćene s krivim identifikatorskim brojem, a oba tipa stranica s greškama su zbog sličnosti spojene u jednu grupu.

U slijedećem koraku smo jednako zastupili sva tri podskupa vrijednosti i rezultat (tablica 5.4) se poboljšao.

Tablica 5.4: Grupiranje s jednako zastupljenim podskupovima

Broj grupa	3	4	5	6	7
Sažetost	0.83113	0.94427	**0.96955**	0.89482	0.95675

Svi elementi su ispravno podijeljeni u svoje grupe uz dodatak da su elementi iz prve skupine stranica, onih ispravnih, s opisom proizvoda podijeljeni u dvije malo različite skupine. To je pokazatelj da je bolje koristiti različite tipove vrijednosti za ispitivanje.

5.3. Grupiranje pomoću sadržaja stranice

Iako je sustav osmišljen da vrši ispitivanje na temelju izgleda stranice i time premosti slučajeve s netipičnim greškama koje trenutni alati traže, pojavili su se slučajevi za koje ta metoda ne funkcionira. Primjer će biti pokazan na temelju stranice čiji je URL prikazan u ispisu 5.8.

Ispis 5.8: URL stranice za ispitivanje

```
http://testasp.acunetix.com/showforum.asp?id=1
```

Modul *crawler* je za spomenutu stranicu našao da za vrijednost parametra *id* koristi vrijednosti od 0 do 5. Za te vrijednosti, aplikacija vraća stranice koje sadrže nekoliko poruka unutar foruma. Ako korisnik unese identifikator koji se ne nalazi u bazi, aplikacija vraća jednostavnu stranicu od svega tri linije teksta s porukom o greški. Zamjenom broja s nekim slovim ili drugim znakom, vraćaju se još dvije poruke o drugim greškama do kojih je došlo. Sve tri stranice prikazane su u ispisima 5.9, 5.10 i 5.11.

Ispis 5.9: Stranica s greškom dobivena uz vrijednost parametra '123456'

```
ADODB.Field error '800a0bcd'

Either BOF or EOF is True, or the current record has been deleted. Requested operation requires a current record.

/showthread.asp, line 10
```

Ispis 5.10: Stranica s greškom dobivena uz vrijednost parametra 'ababab'

```
Microsoft JET Database Engine error '80040e10'

No value given for one or more required parameters.

/showthread.asp, line 9
```

Ispis 5.11: Stranica s greškom dobivena uz vrijednost parametra '[][][]'

```
Microsoft JET Database Engine error '80040e14'

Syntax error (missing operator) in query expression 'a.id = b.forumid AND b.id=[][][]'.

/showthread.asp, line 9
```

Vidimo da tri stranice prikazuju informacije o tri različite greške koje su se dogodile na poslužitelju i htjeli bi da ih naš sustav razvrsta u tri grupe. No, problem je u tome što su sve stranice vrlo slične te postoji svega nekoliko značajki koje im se razlikuju. Posljednja stranica prikazuje grešku koja ukazuje na ranjivost SQL ubacivanja.

Rezultati inicijalnog ispitivanja koje je sadržavalo sve poznate vrijednosti atributa pronađene u *crawleru* te veliki skup vrijednosti koji je trebao dohvatiti sva tri tipa greške je dao rezultate prikazane u tablici 5.5.

Tablica 5.5: Grupiranje

Broj grupa	2	3	4	5
Saženost	0.97332	0.96069	0.59609	0.72452

Usprkos visokom broju srednje sažetosti, rezultat grupiranja nije zadovoljavajući. U slučaju s dvije grupe, sustav je odijelio sve ispravne stranice u jednu grupu i sve stranice s greškama u drugu. To je i bilo za očekivati za taj broj grupa, ali mi smo htjeli odvojiti različite poruke o greškama. Za tri grupe su i dalje sve stranice s greškama ostale u svojoj grupi dok su ispravne podijeljene na temelju internih razlika. Tek pri većem broju grupa su se počele dijeliti i stranice s greškama, ali potpuno neispravno.

Zbog toga je proveden niz testova koji u sebi nije sadržavao ispravne stranice. Početni rezultati su prikazani u prvom retku tablice 5.6. Iz sažetosti se vidi da su uzorci dosta raspršeni što samo po sebi govori o nekim nedostacima. Analizom je utvrđeno da je grupiranje izvedeno uglavnom na temelju značajki dužine stranice i broja riječi, a kako dužina stranice u primjeru greške iz ispisa 5.11 ovisi i o dužini ispitnog niza, grupiranje je uveliko ovisilo o ispitnoj vrijednosti parametra.

Uniformiranjem dužine ispitnih vrijednosti, sažetost je naglo porasla i to je prikazano u drugom primjeru. Iako su naizgled rezultati bolji, utvrđeno je da je uniformiranjem vrijednosti za ispitivanje samo smanjena razlika između jedinih preostalih značajki koje se razlikuju pa je sažetost porasla, dok su greške najpravilnije grupirane u tri grupe, ali to daje manju sažetost od svih ostalih. Za grupiranje na temelju izgleda stranice jednostavno ne postoje značajke koje bi dovoljno dobro opisale razlike. To je zbog toga što razlika u izgledu u principu i nema, već su one samo u tekstu greške.

Tablica 5.6: Sažetosti grupa prilikom ispitivanja

	2 grupe	3 grupe	4 grupe	5 grupa
1.	0.58963	0.72724	0.72081	0.68322
2.	0.92919	0.82251	0.88543	0.90811
3.	0.90937	0.87758	0.89073	0.90997
4.	0.69614	0.98632	0.89295	0.91249

Rješenje koje se nameće je uvođenje značajki koje će opisivati stranicu i na temelju njenog sadržaja. Dodavanjem novih modula u proces izvlačenja značajki dobiveni su rezultati u trećem retku tablice 5.6. Značajke opisuju pojavljivanje riječi, odnosno izraza koje su tipične za svaku od gledanih stranica. Sažetosti su ostale približno jednakim, a to je zbog toga što novouvedene značajke nemaju dovoljnu težinu u ukupnom rezultata.

Dodavanjem težišnih faktora uz značajke koje opisuju sadržaj stranice konačno su postignuti zadovoljavajući rezultati, prikazani u posljednjem retku.

Kroz cijelu ovu analizu došli smo do zaključka da je iznimno bitan broj značajki koje se razlikuju među stranicama. Također je pokazano da je u nekim slučajevima nemoguće vršiti grupiranje samo na temelju izgleda stranice, već da je potrebno uvesti značajke koje opisuju njen sadržaj.

6. Zaključak

U ovom radu su prvo prikazane najkritičnije ranjivosti Web aplikacija i analizirano je kako postojeći alati otvorenog kôda vrše detekciju ranjivosti. Pokazano je da većina detekcije ranjivosti ovisi o detekciji nekih učestalih poruka o greškama ili pak ovisi o pronalaženju vrijednosti ulaznih parametara u odgovorima aplikacije. Mana takvih sustava su aplikacije koje koriste posebno pripremljene poruke o greškama i ne koriste vrijednosti ulaznih parametara u odgovorima pa u tim slučajevima eventualno uzrokovane greške prolaze bez detekcije.

U radu je predložen novi pristup detekciji ranjivosti. Sustav koji implementira navedeni pristup se sastoji od procesa prikupljanja velikog broja stranica i njihovog razvrstavanja u grupe prema sličnosti stranica. Za sličnost stranica pokušalo se promatrati samo izgled stranice kako bi se zaobišli spomenuti nedostatci. U samom početku dizajniranja sustava zaključeno je da sustav neće biti u stanju odrediti grupe za koje smatra da su u njima prisutne stranice s greškama koje mogu ukazivati na ranjivost. Umjesto određivanja konkretne grupe i stranice s potencijalnom ranjivošću, sustav je implementiran na način da korisniku predstavi karakteristične stranice unutar grupiranih razreda. Na taj način se ogromna količina informacija o prikupljenim stranicama smanjuje i korisniku se pokazuje samo mali podskup. Korisnik zatim iz tog podskupa može brzo zaključiti koja grupa predstavlja stranice s greškama i koji su parametri potencijalno ranjivi.

Sustav je izgrađen izrazito modularno i omogućuje veliku manipulaciju s međurezultatima i rezultatima. Usprkos tome, postoji dovoljno prostora za napredak. U modul za prikupljanje informacija o aplikaciji bi trebalo ugraditi parser JavaScript kôda koji bi bio u stanju prikupljati informacije koje se generiraju dinamički ili dohvaćaju preko AJAX zahtjeva. Uz to je prijeko potrebno pojednostavniti način konfiguracije modula za ispitivanje, poželjno pomoću grafičkog sučelja.

Na temelju eksperimenata sa sustavom je zaključeno da je sustav sposoban razlikovati stranice prema njihovom izgledu, slično kao što bi ih korisnik razlikovao. Sustav je također u stanju ocijeniti optimalan broj grupa u koje treba razvrstati stranice. Negativna strana se pokazala u slučajevima kada su pronađene različite poruke o različitim greškama za koje bi htjeli da ih sustav razlikuje, ali su one naizgled toliko slične da sustav to nije u stanju. Tek uvođenjem grupiranja na temelju sadržaja tih poruka i ispravnog odabira vrijednosti za ispitivanje je sustav uspio pravilno grupirati stranice. Iz svega se izvodi zaključak da se sustav može koristiti za detekciju potencijalnih ranjivosti, ali da postoje situacije u kojima neke potencijalne ranjivosti mogu proći bez detekcije.

Osim upotrebe sustava za detekciju ranjivosti, pojedini moduli se mogu iskoristiti u budućnosti i za druge namjene. Tako npr. module za izvlačenje značajki stranica i grupiranje možemo iskoristiti za sve procese gdje nam je potrebno grupirati HTML stranice. Ukoliko je potrebno, modul se može iskoristiti i za grupiranje drugih tipova dokumenata prema odabranom kriteriju sličnosti. Rezultati se također mogu iskoristiti za vizualizaciju grupiranja.

7. Literatura

[1] World Internet Usage Statistics News and Population Stats, dostupno na Internet adresi: http://www.internetworldstats.com/stats.htm

[2] IP Counts by Country, dostupno na Internet adresi: http://www.domaintools.com/internet-statistics/country-ip-counts.html

[3] M. Kozina, Automatizirano određivanje vrste Web aplikacije, 2007.

[4] SANS Institute, dostupno na Internet adresi: http://www.sans.org/

[5] Web Application Security Consortium, dostupno na Internet adresi: http://www.webappsec.org/

[6] CERT, dostupno na Internet adresi: http://www.cert.org/

[7] OWASP, dostupno na Internet adresi: http://www.owasp.org

[8] SecurityFocus, dostupno na Internet adresi: http://www.securityfocus.com/

[9] Common Vulnerabilities and Exposures, dostupno na Internet adresi: http://cve.mitre.org/index.html

[10] S. Christey, R. A. Martin, Vulnerability Type Distributions in CVEVulnerability Type Distributions in CVE, 2007.

[11] OWASP Top 10 2007: The ten most critical Web application security vulnerabilities, 2007

[12] Gamja: Web vulnerability scanner, dostupno na Internet adresi: http://sourceforge.net/projects/gamja

[13] Wapiti, Web application vulnerability scanner, dostupno na Internet adresi: http://wapiti.sourceforge.net/

[14] w3af - Web Application Attack and Audit Framework, dostupno na Internet adresi: http://w3af.sourceforge.net/

[15] Javascript XSS Scanner,, dostupno na Internet adresi: http://www.gnucitizen.org/projects/javascript-xss-scanner/

[16] Web Application Security Consortium – Security Statistics, dostupno na Internet adresi: Statistics,http://www.webappsec.org/projects/statistics/

[17] S. Kamkar, MySpace Worm Explanation, dostupno na Internet adresi: http://namb.la/popular/tech.html

[18] R. Valotta, Nduja Connection Worm, dostupno na Internet adresi: http://rosario.valotta.googlepages.com/

[19] B. Flesch, This is the first Weblog XSS Worm, dostupno na Internet adresi: http://mybeni.rootzilla.de/mybeNi/2007/this_is_the_first_weblog_xss_worm/

[20] SPI Labs, Blind SQL Injection

[21] G.A. Larsen, Using xp_cmdshell, dostupno na Internet adresi: http://www.databasejournal.com/features/mssql/article.php/3372131

[22] SPI Labs, XPath Injection, dostupno na Internet adresi: http://www.webappsec.org/projects/threat/classes/xpath_injection.shtml

[23] S. Faust, LDAP Injection, dostupno na Internet adresi: http://www.webappsec.org/projects/threat/classes/ldap_injection.shtml

[24] A. Klein, HTTP Response Splitting Whitepaper, 2004.

[25] OS commanding, dostupno na Internet adresi: http://www.webappsec.org/projects/threat/classes/os_commanding.shtml

[26] SSI Injection, dostupno na Internet adresi: http://www.webappsec.org/projects/threat/classes/ssi_injection.shtml

[27] Server Side Includes, dostupno na Internet adresi: http://en.wikipedia.org/wiki/Server_Side_Includes

[28] V. A. Diaz, MX Injection - Capturing and Exploiting Hidden Mail Servers, 2006

[29] PHP Includes, dostupno na Internet adresi: http://www.w3schools.com/php/php_includes.asp

[30] GST Assist attack details, dostupno na Internet adresi: http://www.abc.net.au/7.30/stories/s146760.htm

[31] Google, Google Docs, dostupno na Internet adresi: http://docs.google.com

[32] Gmail contact flaw, dostupno na Internet adresi: http://betterexplained.com/articles/gmail-contacts-flaw-overview-and-suggestions/

[33] Rainbow Tables, dostupno na Internet adresi: http://en.wikipedia.org/wiki/Rainbow_table

[34] Google Lab, Google Sets, dostupno na Internet adresi: http://labs.google.com/sets

[35] Demsar J, Zupan B, Leban G, Orange: From Experimental Machine Learning to Interactive Data Mining, 2004.

[36] Python Programming Language, dostupno na Internet adresi: http://www.python.org/

[37] Python Package Index, dostupno na Internet adresi: http://pypi.python.org/pypi

[38] PyXML package, dostupno na Internet adresi: http://pyxml.sourceforge.net/

[39] TidyLib Python wrapper, dostupno na Internet adresi: http://utidylib.berlios.de/

[40] BeautifulSoup package, dostupno na Internet adresi: http://www.crummy.com/software/BeautifulSoup/

[41] ClientCookie package, dostupno na Internet adresi: http://wwwsearch.sourceforge.net/ClientCookie/

[42] C4.5 algorithm, dostupno na Internet adresi: http://en.wikipedia.org/wiki/C4.5_algorithm

[43] A. Jakulin, Data Mining in Python, dostupno na Internet adresi: http://kt.ijs.si/aleks/orng/

[44] XSSDB Baza ranjivosti, dostupno na Internet adresi: http://www.gnucitizen.org/xssdb/

[45] Acunetix, dostupno na Internet adresi: http://www.acunetix.com/

[46] L. Gyergyek, N. Pavešić, S. Ribarić, Uvod u raspoznavanje uzoraka, Tehnička knjiga, Zagreb, 1988.