SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 762

Detekcija korištenih Web tehnologija upotrebom alata Wappalyzer

Adrian Brajković

Zagreb, lipanj 2022.

Umjesto ove stranice umetnite izvornik Vašeg rada.

Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

Zahvaljujem se mentoru doc. dr. sc. Stjepanu Grošu, mag. ing. Ivanu Kovačeviću i Bruni Skendroviću, bacc. ing. comp. na pomoći tijekom izrade ovoga rada.

SADRŽAJ

1. Uvod

Izrada Web stranica zahtjeva tehničke kompetencije i vrijeme koje se u današnje vrijeme sve više naplaćuje. Jedna od popularnih alternativa je korištenje sustava za upravljanje sadržajem, skraćeno CMS (engl. *Content Management System*). Takva rješenja ubrzavaju izradu Web sjedišta i njihova uporaba je mnogima isplativa i pristupačna.

Većina CMS-ova je otvorenog koda koji je dostupan i napadačima. S obzirom na to da imaju pristup kodu, mogu ga proučavati i naći ranjivosti koje onda iskoriste za svoju dobit. Takvi napadi su popularni i mogu prouzročiti velike štete tvrtkama i korisnicima Web stranica.

Važno je što ranije prepoznati ranjive verzije sustava za upravljanje sadržajem i nadograditi na noviju verziju ili popraviti ranjivost. Zato je napravljen servis namijenjen detekciji sustava za upravljanje sadržajem. Za te svrhe se koristi Wappalyzer, alat specijaliziran za detekciju tehnologija koje se koriste na Web stranicama. Istražene su tehnike koje Wappalyzer koristi i primijenjene su na ovaj program.

2. Opis sustava

Glavna zadaća servisa je procesiranje podataka o Web stranicama *hr* domene. Podaci se nalaze u bazi na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave (ZEMRIS) Fakulteta elektrotehnike i računarstva. Procesiranje i analiza podataka daju uvid u tehnologije koje pojedina Web stranica koristi. Ovaj sustav se fokusira na tehnologije koje pripadaju sustavima za upravljanje sadržajem. Detekciju sustava za upravljanje sadržajem i ostalih tehnologija obavlja alat Wappalyzer. Posebna pažnja je posvećena onim Web stranicama iz kojih se može prepoznati verzija sustava za upravljanje sadržajem koju koristi. Ta nam informacija daje više detalja o sigurnosnim aspektima Web stranice.

Servis je namijenjen da se pokreće periodički, tako da se analiziraju novi podaci iz baze. Sljedeći korak i cilj kojim se ovaj rad neće baviti je uspoređivanje rezultata s bazama ranjivih verzija sustava za upravljanje sadržajem. Za to veliku ulogu igra određivanje verzija sustava za upravljanje sadržajem.

2.1. Sustavi za upravljanje sadržajem

Veliki broj Web aplikacija koriste gotova rješenja u obliku sustava za upravljanje sadržajem [1]. Takvi sustavi omogućuju korisnicima koji nemaju opširno znanje programiranja i razvoja softvera da lako izrade Web stranicu. Nudi im se širok spektar mogućnosti prilagodbe Web stranice njihovim zahtjevima.

Posebno se ističu sustavi za upravljanje sadržajem koji su otvorenog koda i besplatni za korištenje. Oni su prihvatljivi manjim poduzećima zbog niskih troškova razvoja Web stranica. S rastom popularnosti raste i opasnost od napadača koji iskorištavaju ranjivosti tih Web stranica. Često su ti napadi potpuno automatizirani i mnogi alati omogućuju početnicima iskorištavanje ranjivosti. Sve to čini sustave za upravljanje sadržajem glavnim izvorom sigurnosnih ranjivost na Internetu [1].

Dovoljno je da napadač otkrije jednu ranjivost u nekom sustavu za upravljanje sadržajem. Postoji više stranica koje koriste isti taj sustav za upravljanje sadržajem i svaka od njih je moguća meta napadača jer imaju istu ranjivost. Unatoč osviještenosti o prijetnjama i ranjivostima, sigurnost sustava za upravljanje sadržajem ostaje nejasna.

2.1.1. WordPress

Na tržištu je dostupna golema količina sustava za upravljanje sadržajem, ali među njima se posebno ističe jedan. WordPress je nastao 2003. godine iz platforme za blogove, b2, a danas je jedan od najpopularnijih alata za stvaranje Web stranica [2]. Brojke pokazuju i njegovu dominaciju nad ostalima sustavima za upravljanje sadržajem, više od 60% Web stranica koje koriste neki CMS koriste WordPress [2]. Napisan je u PHP programskom jeziku i koristi MySQL bazu podataka. Od početaka je cilj WordPressa bio da svaka osoba bez tehničkog predznanja može stvoriti i objaviti Web stranicu. Prema W3Techs [3], 42.9% Web stranica na internetu koristi WordPress. Sljedeći sustav za upravljanje sadržajem po popularnosti je Shopify, kojeg koristi 4.3% Web stranica.

Najveća negativna strana WordPressa je da je ujedno i popularna meta među kibernetičkim kriminalcima [4]. WordPress dolazi s repozitorijem od nekoliko desetaka tisuća dodataka i tema. Oni poboljšavaju funkcionalnost Web stranice na različite načine i pozitivno utječu na korisničko iskustvo. Također su i najslabija karika u ovom ekosustavu zbog velikih sigurnosnih propusta u nekima od njih. WordPress dodaci s greškama u kodu postaju glavni ulaz u sustav kojeg iskorištavaju napadači [4]. Iako je najbolji način za zaštitu Web stranica korištenje pouzdanih dodataka i redovna nadogradnja na najnovije verzije, puno Web stranica ne prati ove upute. Sigurnosni nedostaci su razlog zašto su WordPress stranice jedne od najpopularnijih meta za napadače.

2.2. Ranjivosti sustava za upravljanje sadržajem

Prema istraživanju iz 2013. godine, 73.2% najpopularnijih WordPress stranica je koristilo ranjivu verziju WordPressa [5]. Istraživanje je provedeno nad 42,106 stranica. Neke od tih ranjivosti se mogu detektirati besplatnim automatiziranim alatima. Ti alati mogu iskoristiti ranjivosti u nekoliko minuta.

Neki od popularnijih napada na sustave za upravljanje sadržajem su:

- Napad grubom silom

- Napad ubacivanjem SQL izraza
- Cross-Site Scripting (XSS)
- Napad uskraćivanja usluge
- Napad prolaska direktorijima

2.2.1. Napad grubom silom

Napad grubom silom koristi metodu pokušaja i pogrešaka prilikom upisivanja korisničkih imena i zaporki sve dok ne dođe do uspješne kombinacije. Zadana postavka u WordPressu je da ne ograničava broj pokušaja prijave u sustav. Takav sigurnosni propust ostavlja korisnike s jednostavnim zaporkama ranjivima na napade grubom silom. Čak i ako napad ne uspije, količina zahtjeva koji se generira prilikom napada može uzrokovati pad dostupnosti poslužitelja.

Cilj napada grubom silom na WordPress stranice je pribaviti točnu zaporku za administratorski korisnički račun. Time napadač dobije pristup administratorskoj kontrolnoj ploči i kontrolu nad cijelom Web stranicom. Ranijim verzijama WordPressa je zadano bilo korisničko ime *admin* za administratora stranice [6]. Većina napada grubom silom pretpostavlja da administratori još uvijek koriste to korisničko ime. To napad čini lakšim za provesti jer više nije potrebno pogoditi i korisničko ime, već samo zaporku.

2.2.2. Napad ubacivanjem SQL izraza

Napad ubacivanjem SQL izraza je jedan od najrazornijih ranjivosti koje se mogu dogoditi sustavu [7]. Može dovesti do izlaganja osjetljivih informacija iz baze podataka, uključujući korisnička imena, zaporke, brojevi mobitela, adrese i podaci o bankovnim karticama. Ubacivanje SQL izraza omogućuje napadaču manipulaciju SQL kodom, čime mijenja predviđenu funkcionalnost.

Na primjeru jednostavnog isječka PHP koda može se demonstrirati ranjivost i kako se ona iskorištava.

```
$input = $_POST["email"];
$sql = "SELECT * FROM Users WHERE email = '$input';";
```

Ispis 2.1: Isječak programskog koda ranjivog na napad umetanjem SQL izraza

SQL iz Ispisa 2.1 se gradi uz pomoć podataka o elektroničkoj pošti poslanih s klijentske strane. Ono što napadač može napraviti je poslati sljedeći ulaz umjesto valjane elektroničke pošte:

```
user@gmail.com' OR '1'='1
```

Sljedeći SQL kod će se izvesti u bazi podataka:

```
SELECT * FROM Users WHERE email = 'user@gmail.com' OR '1'='1';
```

Izraz je istinit za svaki redak u tablici Users i to dovodi do curenja informacija o svim korisnima napadnute Web stranice.

Osim curenja podataka, napadom ubacivanja SQL izraza je moguće i izbrisati tablice iz baze podataka, urediti postojeće podatke u bazi ili unijeti nove podatke. Takve ranjivosti nastaju kad programer ne validira ulaz poslan s korisničke strane prije nego ga proslijedi bazi podataka. Prema web stranici iThemes, u 2021. godini je 9.3% ranjivosti u WordPressu bilo ubacivanje SQL izraza [8].

Dobra praksa za obranu od takvih napada je korištenje pripremljenih izraza (engl. *prepared statement*). Korištenje pripremljenih izraza onemogućava izvršavanje korisničkog unosa kao SQL koda, već se umeće poslije prevođenja i ponaša kao običan niz znakova [9].

2.2.3. Cross-Site Scripting (XSS)

Na sličnom principu funkcioniraju i Cross-Site Scripting (XSS) napadi. Oni spadaju u napade ubacivanja, točnije, ubacuju se zloćudne skripte u inače vjerodostojne stranice. Najčešće su u obliku JavaScript koda, a žrtve su korisnici kojima se pošalje zaraženi link.

Slijedi demonstracija ranjivog koda u programskog jeziku PHP te njegovog iskorištavanja za XSS napad. Neka Web stranica http://trusted-site.com vodi do sljedećeg koda:

```php
<?php
  $name = $_GET["name"];
  echo "<p>Hello $name</p>";
?>
```

Ispis 2.2: Poslužiteljski kod ranjiv na XSS napad

Ispis 2.2 pokazuje kako kroz parametre u URLu se pošalje ulaz kojeg poslužitelj ne provjerava, već ga ispisuje kao HTML kod. Zbog toga se na stranicu može ubaciti proizvoljan JavaScript kod, a on ima pristup kolačićima u pregledniku koji mogu sadržavati povjerljive podatke poput identifikatora sjednice na stranici. Krađom tog podatka napadač se može predstavljati kao druga osoba.

Primjer URLa koji iskorištava ranjivost:

http://trusted-site.com?name=<script>...</script>

Unutar *script* elemenata napadač ubacuje svoj zloćudan kod i takvu poveznicu šalje potencijalnim žrtvama. Napadač može pomoću maliciozne skripte ukrasti podatke iz preglednika poput kolačića i proslijediti ih na drugi poslužitelj kojem ima pristup.

Najlakši način za obranu od XSS napada je da se sav ulaz od korisnika predznači (engl. *escape*), točnije da se predznače specijalni znakovi unutar korisničkog ulaza. Kao primjer će se iskoristiti ranjivi kod iz Ispisa 2.2 i modificirati tako da se ubačena maliciozna skripta tretira kao običan niz znakova.

```
<?php
  $name = htmlentities($_GET["name"]);
  echo "<p>Hello $name</p>";
?>
```

Ispis 2.3: Predznačivanje specijalnih znakova u ulazu

Ispis 2.3 prikazuje kako zaštititi programski kod iz Ispisa 2.2 od XSS napada. Funkcija *htmlentities* obavlja predznačivanje specijalnih znakova i time sprječava izvođenje skripte. U većini modernih radnih okruženja za razvoj Web aplikacija to se događa automatski i programer ne treba voditi brigu o takvim ranjivostima.

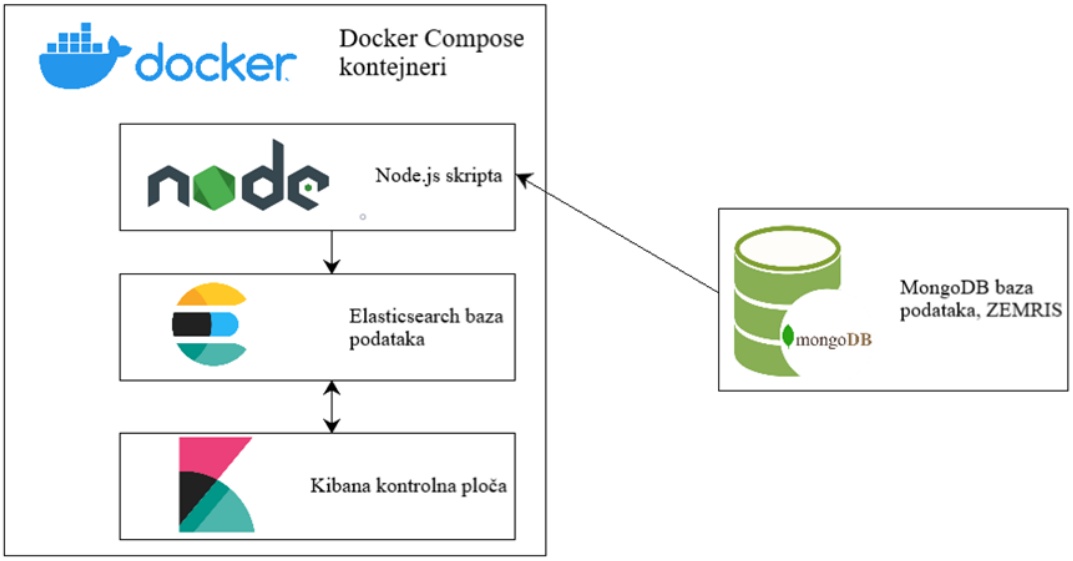
Prema CVE Details bazi sigurnosnih ranjivosti, XSS ranjivosti čine 35.8% svih WordPress ranjivosti [10]. Nisu svi sustavi za upravljanje sadržajem jednako ranjivi na sve napade, ali ti su napadi još uvijek prisutni. Popularni sustavi za upravljanje sadržajem su meta 20% svih zloćudnih aktivnosti [11]. Iako neka poduzeća provode penetracijske testove kako bi identificirali sigurnosne propuste, većini njih to nije praksa.

Svi ti napadi i ranjivosti su razlog zašto se sustavi za upravljanje sadržajem često smatraju nesigurnima. Cilj ovog završnog rada i napisanog programskog rješenja uz rad je prepoznati koje stranice koriste sustave za upravljanje sadržajem i koje verzije sustava. To nam može omogućiti detekciju nesigurnih i ranjivih Web stranica ako se dobiveni podaci usporede s poznatim napadima na određene verzije sustava za upravljanje sadržajem.

3. Arhitektura sustava

Servis se sastoji od tri dijela koji su prikazani kao Docker kontejneri u Slici 3.1

- Node.js skripta
- Elasticsearch baza podataka
- Kibana kontrolna ploča



Slika 3.1: Prikaz arhitekture sustava

Slika 3.1 prikazuje i tok podataka između određenih servisa. U skripti se nalazi cijela logika koja dohvaća sirove podatke iz MongoDB baze, parsira ih i u odgovarajućem obliku šalje Wappalyzer alatu na analizu. Rezultati analize se filtriraju tako da ostanu samo detekcije sustava za upravljanje sadržajem. Ti se rezultati pohranjuju u Elasticsearch bazu podataka. Za vizualizaciju i pregled podatka iz Elasticsearch baze koristi se Kibana kontrolna ploča.

Cijeli sustav se pokreće u Docker kontejnerima. Elasticsearch, Kibana i Node.js se nalaze u svojim kontejnerima. Detaljnije specifikacije se nalaze u *Dockerfile* i *docker-compose.yml* datotekama.

3.1. Node.js

Node.js [12] je izvršno okruženje koje se koristi za pokretanje JavaScript koda na poslužitelju, izvan Web preglednika. Omogućuje izradu dinamičkih Web stranica i alata za pokretanju u naredbenom retku, a dizajniran je za gradnju skalabilnih mrežnih aplikacija. Node.js je asinkron i vođen događajima što znači da prilikom izlazno ulaznih operacija poput HTTP poziva, čitanja iz baze ili diska, program ne čeka na odgovor, već nastavlja s radom ostalog dijela. Tek nakon što dobije odgovor će nastaviti s operacijama nad tim podacima. Node.js se pokreće u jednom procesu, bez stvaranja nove dretve za svaki zahtjev. To omogućuje tisuće istodobnih veza s jednim poslužiteljem bez potreba za ručnim upravljanjem dretvama.

Node.js koristi V8 JavaScript engine što mu daje dobre performanse. Popularnosti Node.js-a pridonosi i Node package manager (npm), upravitelj paketa koji sadrži više od milijun paketa otvorenog koda.

U ovom sustavu je Node.js servis središnji dio u kojem se odvija cijela logika servisa, dohvaćanje podataka iz MongoDB baze, parsiranje i sanitizacija podataka te naposljetku pokretanje Wappalyzer analize i konačno spremanje rezultata u Elasticsearch.

3.2. MongoDB

MongoDB [13] je NoSQL baza podataka u kojoj su zapisi pohranjeni kao dokumenti poput JSON objekata. Svaki dokument se nalazi unutar kolekcije, a baza se sastoji od kolekcija. Kolekcije su analogne tablicama u relacijskim bazama podatka.

Prednost korištenja dokumenata je u tome što tipovi podataka u dokumentu korespondiraju onim tipovima koji se koriste u programskim jezicima. Dokumenti kao vrijednost nekog polja mogu imati drugi dokument što smanjuje potrebu za skupim operacijama *join* koje su karakteristične za relacijske baze poput PostgreSQL-a. Struktura dokumenta nije fiksna kao kod relacijskih baza, već je dinamička.

MongoDB baza na ZEMRIS-u sadrži podatke o Web stranicama *hr* domene. Neki od tih podataka se koriste za analizu i detekciju sustava za upravljanje sadržajem što će detaljnije biti pojašnjeno u poglavlju *Node.js skripta*.

3.3. Elasticsearch i Kibana

MongoDB i Elasticsearch [14] su slični po načinu na koji spremaju podatke. Koriste dokumente za pohranu. Oni su u JSON obliku, svaki ima kolekciju ključeva s pripadajućim vrijednostima. Razlika je u njihovoj namjeni, Elasticsearch je namijenjen da se koristi kao tražilica i alat za analitiku podataka. S druge strane, MongoDB je namijenjen da bude skladište podataka.

Neke od primjena Elasticsearch baze su:

- Pretraživanje aplikacija
- Pretraživanje Web stranica
- Sigurnosna analitika
- Poslovna analitika
- Kontrola performansi aplikacija

Elasticsearch koristi invertirani indeks, strukturu podataka koja je dizajnirana da podrži vrlo brze upite nad tekstovima. Svaki uneseni dokument se indeksira što omogućuje pretragu u gotovo stvarnom vremenu.

U ovom sustavu, Elasticsearch se koristi za pohranjivanje podataka o uspješnim detekcijama sustava za upravljanje sadržajem. Ti podaci se kasnije koriste za statistike, a za to je namijenjena Kibana.

Kibana [15] je klijentska aplikacija koja pruža opcije pretrage i vizualizacija podataka nad indeksima u Elasticsearchu. Nudi opcije kreiranja grafikona, tablica, histograma i mapa nad podacima.

Slika 3.2 prikazuje kako Kibana vizualizira podatke iz Elasticsearch indeksa. U ovom primjeru su odabrana samo polja *main_url*, *cms_name* i *cms_version* zbog jednostavnosti. Kibana omogućuje prikaz proizvoljnih polja i filtriranje koristeći upite.

	main_url	cms_name	cms_version
›	https://wp.scandieuro.hr	Joomla	1.5
›	https://pepsi.hr	Sitefinity	10.2.6653.0 PE
›	https://www.ozimec.hr	Joomla	1.5
›	https://khanzic.webnode.hr	WebNode	2
›	https://www.kkkzagreb.hr	Joomla	1.5
›	https://www.tomato.com.hr	Liferay	7.0.10
›	https://www.jadrolinija.hr	Sitefinity	6.2.4910.0 SE

Slika 3.2: Pregled rezultata Wappalyzer analize u Kibani

3.4. Docker

Docker [16] je alat koji omogućuje jednostavno upogonjavanje aplikacija i sustava u zaštićenim okruženjima, kontejnerima. Kontejneri sadrže sve zavisnosti potrebne aplikaciji pa se gubi potreba za instalacijom dodatnih alata na računalu ili poslužitelju gdje se Docker kontejner pokreće. Zbog toga je programerima omogućeno da uvijek razvijaju aplikaciju u istom okruženju, bez obzira na vrstu računala ili operacijski sustav i programe koje koriste za rad.

Docker kontejneri su slični virtualnim strojevima, ali njihova prednost je u tome što zahtijevaju manje računalnih resursa. Kontejneri sadrže samo ono što je potrebno za alat aplikacije, točnije samo ono što je programer napisao da bude u kontejneru. S druge strane, virtualni strojevi sadrže cijeli operacijski sustav koji zahtjeva više računalne snage.

Kontejneri su dostupni na internetu, a postoji i mogućnosti kreiranja svojeg. Tada se stvaraju posebne *Dockerfile* datoteke u kojima se podešava sve potrebno za pokretanje Docker kontejnera. Svaka instrukcija u datoteci stvara jedan sloj, a kad se neka instrukcija promijeni, onda pri pokretanju kontejnera se izmjeni samo taj sloj, što čini Docker brzim.

Cijeli sustav za detekciju sustava za upravljanje sadržajem se nalazi u Docker kontejnerima, a s obzirom na to da su potrebna tri kontejnera, po jedan za Node.js skriptu, Elasticsearch i Kibanu, potrebno je bilo koristi alat Compose.

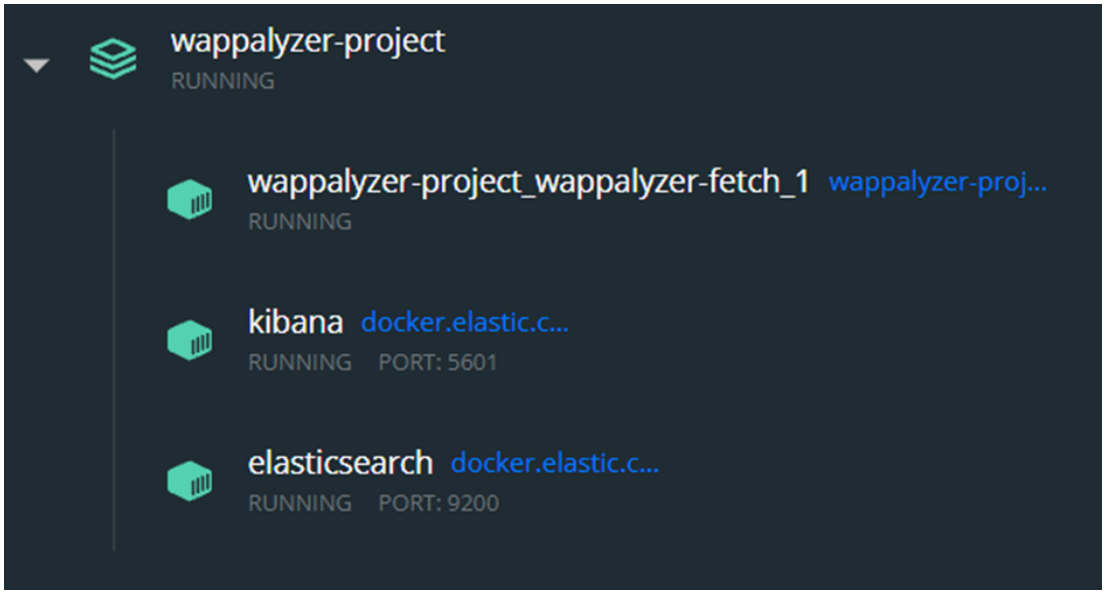
3.4.1. Docker Compose

Compose [17] je alat koji omogućuje pokretanje Docker aplikacija s više kontejnera. Za konfiguraciju kontejnera se koristi *docker-compose.yml* datoteka. U njoj se definiraju servisi potrebni za rad sustava. Svaki od servisa može biti neki javno dostupni Docker kontejner, a može i za svaki servis napisati zasebna *Dockerfile* datoteka. To omogućuje pokretanje sustava jednom naredbom, a Docker se brine o tome kako će se svaki servis pokrenuti.

U ovom sustavu je Docker Compose korišten za definiranje arhitekture sustava, gdje je kontejner za Node.js ručno definiran u *Dockerfileu*, a za Elasticsearch i Kibanu su korištena gotova rješenja.

Raspored kontejnera je vidljiv unutar Docker Desktop aplikacije prilikom pokretanja. Na Slici 3.3 su prikazani Docker kontejneri. Svaki od njih ima naziv, status i port na

kojem su dostupni. Vidljivo je da su Kibana i Elasticsearch dostupni na *portovima* 5601 i 9200. Status svakog kontejnera je *running* što znači da su aktivni i obavljaju svoje funkcije.



Slika 3.3: Prikaz Docker Compose kontejnera u aplikaciji

4. Wappalyzer

Wappalyzer [18] je alat koji se koristi za detekciju tehnologija koje Web stranice koriste. Svaka tehnologija je smještena u neku kategoriju. U trenutku pisanja ovog rada, Wappalyzer u svojoj bazi ima 2845 Web tehnologija raspodijeljenih u 106 kategorija.

U Tablici 4.1 su prikazane neke od kategorija koje Wappalyzer koristi za grupiranje tehnologija. Za svaku kategoriju je prikazano nekoliko tehnologija. U kategoriji sustava za upravljanje sadržajem WordPress, Joomla i Drupal su samo neke od tehnologija koje Wappalyzer može detektirati na Web stranicama.

Kategorija	Tehnologije
Sustavi za upravljanje sadržajem	WordPress, Joomla, Drupal, Craft CMS
JavaScript razvojni okviri	React, Angular, Vue.js, InfernoJS, Alpine.js
Web razvojni okviri	Django, Symfony, Laravel, Ruby on Rails, Next.js, Spring
Web poslužitelji	Apache, XAMPP, Nginx, Oracle HTTP Server
Baze podataka	PostgreSQL, MongoDB, MySQL, Firebase, Redis, SQLite
Operacijski sustavi	Ubuntu, Windows Server, Debian, CentOS, Alpine Linux

Tablica 4.1: Primjeri kategorija i tehnologija koje Wappalyzer može detektirati

Wappalyzer prepoznaje uzorke specifične za određenu tehnologiju. Koristi regularne izraze nad javnim podacima s Web stranice i HTTP odgovora kako bi prepoznao koje tehnologije Web stranica koristi.

Wappalyzer koristi sljedeće podatke tijekom analize Web stranica:

- Kolačići
- HTML/CSS/JavaScript izvorni kod

- Zaglavlja HTTP odgovora
- URL stranice
- Meta podaci
- DNS zapisi
- Sadržaj robots.txt datoteke

Nakon izvršene analize, Wappalyzer vraća podatke o svakoj detektiranoj tehnologiji. To su ime tehnologije, sigurnost (engl. *confidence*) detekcije tehnologije (0-100%), verzija koja može biti i *null* ako nije detektirana, Web stranica tehnologije i popis kategorija kojima tehnologija pripada.

4.1. Wappalyzer usluge

Izvorni kod Wappalyzera je javno dostupan, a za ovaj rad je korištena JavaScript verzija. U repozitoriju se nalazi i popis svih tehnologija s pravilima detekcije. Osim toga, Wappalyzer nudi SaaS (softver kao usluga, engl. software as a service) rješenja koje se plaćaju, ali nude i više opcija od samostalnog korištenja alata.

Javno dostupan izvorni kod može samo detektirati tehnologije koje koristi Web stranica. Ovaj rad koristi javno dostupnu verziju Wappalyzera, bez dodatnih plaćenih usluga. MongoDB baza na ZEMRIS-u sadrži sve potrebne podatke o Web stranicama za efikasnu detekciju sustava za upravljanje sadržajem. Kompletna analiza se odvija lokalno, bez potrebe za pozivima prema Wappalyzer APIju.

Wappalyzer nudi i proširenja za Internet preglednike Chrome, Firefox, Edge i Safari. Proširenje omogućuje korisnicima da u trenutku posjete Web stranici saznaju koje tehnologije ona koristi. Proširenje za Gmail koristi domenu elektroničke pošte pošiljatelja kako bi detektirao tehnologije koje koristi njihova Web stranica. Osim toga, prikazuje i kontakt podatke poput telefonskih brojeva i profila s društvenih mreža.

4.2. Pravila detekcije

Popis tehnologija i regularnih izraza koji se koriste za detekciju je javno dostupan i nalazi se u GitHub repozitoriju Wappalyzera. Spremljeni su u JSON datotekama i korisnik ima mogućnost ubaciti svoja pravila detekcije ili filtrirati tehnologije po svojim potrebama.

Na primjeru isječka dokumenta s WordPress pravilima pokazat će se kako funkcionira

Wappalyzer.

Slika 4.1 prikazuje neke od načina na koji se WordPress može detektirati na Web stranicama. Svaki par ključa i vrijednosti će biti opisan u sljedećim poglavljima.

```
{
    "headers": {
      "X-Pingback": "/xmlrpc\\.php$",
      "link": "rel=\"https://api\\.w\\.org/\""
    },
    "html": [
      "<link rel=[\"']stylesheet[\"'] [^>]+/wp-(?:content|includes)/",
      "<link[^>]+s\\d+\\.wp\\.com"
    ],
    "meta": {
      "generator": "^WordPress(?: ([\\d.]+))?\\;version:\\1",
      "shareaholic:wp_version": ""
    },
    "scriptSrc": [
      "/wp-(?:content|includes)/",
      "wp-embed\\.min\\.js"
    ]
}
```

Slika 4.1: Isječak unosa za WordPress u popisu tehnologija

4.2.1. Zaglavlje HTTP odgovora

Prvi ključ headers iz Ispisa 4.1 sadrži pravila kojima Wappalyzer preko HTTP zaglavlja detektira WordPress kao tehnologiju koju koristi Web stranica. Ako HTTP odgovor s Web stranice sadrži *X-Pingback* zaglavlje koje završava s */xmlrpc.php* ili zaglavlje *link* koje sadrži *rel="https://api.w.org/"*, Wappalyzer identificira WordPress.

Ispis 4.1 prikazuje dva zaglavlja dobivena slanjem HTTP zahtjeva Web stranici. Koristi se alat *curl*.

```
curl -v https://jquery.com/
```

```
x-pingback: https://jquery.com/xmlrpc.php
link: <//jquery.com/wp-json/>; rel=https://api.w.org
```

Ispis 4.1: Isječak HTTP zaglavlja

Oba zaglavlja iz 4.1 zadovoljavaju pravila iz Slike 4.1 i zaključujemo da *https://jquery.com/* koristi WordPress. Wappalyzeru bi bilo dovoljno da je postojalo samo jedno zaglavlje, a moguće je da ispravnih zaglavlja za detekciju nema, već se tehnologija detektira koristeći druge stavke Web stranice, poput HTML koda.

4.2.2. HTML izvorni kod

Polje *html* iz Slike 4.1 sadrži regularne izraze kojima se detektira WordPress iz izvornog koda stranice. Korištenjem opcije *Pogledaj izvor stranice* u Google Chrome pregledniku prikazuje se HTML sadržaj Web stranice. Za demonstraciju će se koristiti Web stranica *https://jquery.com/*.

Ispis 4.2 prikazuje neke *link* elemente iz Web stranice. Svaki od tri *link* elementa iz Ispisa 4.2 se podudara s regularnim izrazom za detekciju WordPressa iz HTML sadržaja Web stranice. Uspješno podudaranje znači da je detektirana tehnologija.

```
<link rel="stylesheet" href="/jquery-wp-content/themes/jquery/css/base.css?v=3">
<link rel="stylesheet" href="/jquery-wp-content/themes/jquery.com/style.css?v=2">
<link rel="stylesheet" href="/jquery-wp-content/themes/jquery/css/docsearch.css">
```

Ispis 4.2: Isječak izvornog HTML koda s *https://jquery.com/*

4.2.3. Meta podaci

U slučaju WordPressa, meta podaci su jedini način za detektirati i verziju WordPressa koju stranica koristi. Ti podaci se nalaze u HTML kodu stranice, u *meta* elementima. Zbog načina na koji Wappalyzer funkcionira, potrebno je te podatke izvući iz koda i pretvoriti u poseban JSON oblik, o tome više u poglavlju *Prilagođavanje podataka iz MongoDB baze*.

Korištenjem opcije *Pogledaj izvor stranice* u Google Chrome pregledniku nad stranicom *https://jquery.com/* mogu se izdvojiti sljedeći meta elementi:

```
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="author" content="JS Foundation - js.foundation">
<meta name="description" content="jQuery: The Write Less, Do More">
<meta name="viewport" content="width=device-width">
<meta name="generator" content="WordPress 4.5.2" />
```

Ispis 4.3: Meta podaci s *https://jquery.com/*

Ispis 4.3 prikazuje *meta* elemente iz HTML koda Web stranice. Korisni su oni *meta* elementi koji sadrže atribute *name* i *content*. Vrijednost atributa *name* se preslikava u ključ unutar JSON objekta, a vrijednost atributa *content* se preslikava u vrijednost za navedeni ključ.

Za sljedeći element

```
<meta name="generator" content="WordPress 4.5.2"/>
```

generira se JSON objekt:

```
{
    generator: "WordPress 4.5.2"
}
```

Ispis 4.4: Meta podaci u obliku JSON objekta

Objekt iz Ispisa 4.4 zadovoljava pravilo za WordPress i time je i detektirana verzija 4.5.2 na stranici *https://jquery.com/*.

4.2.4. Nazivi JavaScript datoteka

Wappalyzer može detektirati tehnologije iz podataka o vanjskim JavaScript datotekama koje Web stranica koristi. Te se datoteke uključuju u Web stranicu koristeći *script* elemente. Zato nije dovoljno Wappalyzeru poslati samo HTML sadržaj stranice, već je potrebno parsirati stranicu i izvući veze do tih datoteka. Više o tome u poglavlju *Dohvat sadržaja Web stranica*.

Na Slici 4.1 pod ključem *scriptSrc* (skraćeno od *script source*) se nalazi popis regularnih izraza. Ako naziv JavaScript datoteka zadovolji neki od navedenih regularnih izraza, Wappalyzer označuje WordPress kao jednu od tehnologija koju Web stranica koristi.

Prvi korak je izdvojiti *script* elemente s Web stranice. Kao primjer će se koristiti Web stranica *https://jquery.com/*.

```
<script src="/jquery-wp-content/themes/jquery/js/modernizr.custom.2.8.3.min.js"/>
<script src="https://code.jquery.com/jquery-1.11.3.js"></script>
<script src="/jquery-wp-content/themes/jquery/js/plugins.js"></script>
<script src="/jquery-wp-content/themes/jquery/js/main.js"></script>
<script src="https://use.typekit.net/wdelaof.js"></script>
<script type='text/javascript' src='/wp-includes/js/wp-embed.min.js?ver=4.5.2'/>
```

Ispis 4.5: Odabrani *script* elementi s *https://jquery.com/*

Ako se izdvoji sadržaj *src* atributa iz Ispisa 4.5 i prikaže se u obliku niza prilagođenog za Wappalyzer analizu, dobije se sljedeći ulaz za Wappalyzer:

```
[
    /jquery-wp-content/themes/jquery/js/modernizr.custom.2.8.3.min.js
    https://code.jquery.com/jquery-1.11.3.js
    /jquery-wp-content/themes/jquery/js/plugins.js
    /jquery-wp-content/themes/jquery/js/main.js
    https://use.typekit.net/wdelaof.js
    /wp-includes/js/wp-embed.min.js?ver=4.5.2
]
```

Ispis 4.6: Izdvojene veze do JavaScript datoteka iz Ispisa 4.5

Jasnije razrađeni regularni izrazi iz Slike 4.1 po ključem *scriptSrc*:

- Naziv JavaScript datoteke sadrži */wp-content/* ili */wp-includes/*
- Naziv JavaScript datoteke sadrži *wp-embed.min.js*

Iz Ispisa 4.6 možemo izdvojiti posljednju JavaScript datoteku koja zadovoljava oba uvjeta. Wappalyzer i na temelju JavaScript datoteka zaključuje da Web stranica koristi tehnologiju WordPress.

4.3. Detekcije neovisne o regularnim izrazima

Pravila definirana regularnim izrazima nisu jedini način kojim Wappalyzer identificira tehnologije korištene na Web stranicama. Postoje tehnologije koje ne moraju biti izravno detektirane preko regularnih izraza, već podrazumijevano dolaze uz drugu tehnologiju. Jednostavni primjer je WordPress za koji je poznato da je napisan u programskom jeziku PHP i koristi MySQL bazu podataka. Wappalyzer i bez izravne detekcije tih tehnologija zaključuje da ih Web stranice koriste ako je detektiran WordPress.

Taj podatak se u bazi tehnologija nalazi u polju *implies*. To znači da prisutnost jedne tehnologije može implicirati prisutnost drugih. Na primjeru zapisa za WordPress će se pokazati kako je to prikazano u Wappalyzer bazi. Ispis 4.7 prikazuje kako Wappalyzer prikazuje da postojanje tehnologije WordPress implicira i korištenje programskog jezika PHP te MySQL baze podataka.

```
"implies": [
    "PHP",
    "MySQL"
]
```

Ispis 4.7: Isječak dokumenta o WordPressu iz baze, polje *implies*

Suprotno tome je polje *excludes*. Ono označava da prisutnost jedne tehnologije isključuje mogućnost neke druge tehnologije na toj Web stranici.

Polje *requires* označava uvjet koji je potrebno ispuniti kako bi detekcija prošla. Taj uvjet je prisutnost neke druge tehnologije. Dodaci za WordPress su primjer gdje je dobro koristiti polje *requires*. Nema smisla detektirati neki WordPress dodatak ako prije toga nije detektiran sam WordPress na Web stranici.

Za primjer će se uzeti W3 Total Cache, dodatak za WordPress koji poboljšava vidljivost Web stranice na tražilicama i smanjuje vrijeme učitavanja Web stranica koristeći *cache*.

Isječak dokumenta za W3 Total Cache iz baze tehnologija:

```
"W3 Total Cache": {
    "requires": "WordPress"
}
```

Ispis 4.8: Polje *requires* iz dokumenta za W3 Total Cache

Ispis 4.8 prikazuje kako Wappalyzer u bazi tehnologija zapisuje uvjet *requires* za detekciju tehnologija.

Polje *requiresCategory* funkcionira na istom principu kao i polje *requires*, ali u sebi sadrži popis kategorija. Detekcija prolazi samo ako prije toga već detektirana tehnologija koja se nalazi pripada zahtijevanoj kategoriji.

Svi opisani načini detekcije tehnologija su korišteni u ovom sustavu. Ako se podaci iz MongoDB baze dobro iskoriste, Wappalyzer postaje moćan alat za detekciju tehno-

logija na Web stranicama. Nakon opisa kako interno funkcionira Wappalyzer, slijedi poglavlje o tijeku programa i kako je Wappalyzer integriran u sustav.

5. Tijek programa

U ovom poglavlju će biti detaljno opisan tijek programa, od pokretanja pa sve do zapisa rezultata u Elasticsearch. Za pokretanje servisa je potrebno instalirati Docker i preusmjeriti lokalni *port* 27017 na IP adresu 161.53.65.4 na kojoj se nalazi poslužitelj s MongoDB bazom. Potrebno je i imati korisničko ime i zaporku za pristup bazi.

Preusmjeravanje se vrši izvođenjem sljedeće naredbe (za korisničko ime *abrajkovic*):

```
ssh -v -NL 27017:127.0.0.1:27017 abrajkovic@161.53.65.4
```

i upisivanjem zaporke.

5.1. Pokretanje Docker kontejnera

Izvođenjem naredbe

```
docker-compose up --build
```

unutar repozitorija se pokreću Docker kontejneri. Postavke kontejnera se definiraju u datoteci *docker-compose.yml*. U Ispisu 5.1 se pod ključem *services* definiraju kontejneri koji će se koristiti. U slučaju ovog sustava, to su:

- *elasticsearch*, baza podataka
- *kibana*, kontrolna ploča
- *wappalyzer-fetch*, glavni dio sustava sa skriptom

```
services:
  elasticsearch:
    image: "docker.elastic.co/elasticsearch/elasticsearch:7.13.3"
    ports:
      - '9200:9200'
  kibana:
    image: "docker.elastic.co/kibana/kibana:7.13.3"
    ports:
      - '5601:5601'
    environment:
      ELASTICSEARCH_HOSTS: '["http://elasticsearch:9200"]'
  wappalyzer-fetch:
    build:
      dockerfile: Dockerfile
      context: .
```

Ispis 5.1: Izdvojeni dijelovi *docker-compose.yml* datoteke

Za *elasticsearch* i *kibana* kontejnere se definira slika kontejnera (engl. *container image*) koja se koristi. To su službene slike za Elasticsearch i Kibanu. Odabrana je verzija 7.13.3 kako bi se podudarala s verzijom na poslužitelju na ZEMRIS-u.

Za Elasticsearch i Kibana kontejnere se definira *port* na kojem su dostupni. Koriste se zadani *portovi*, 9200 za Elasticsearch i 5601 za Kibanu. Kibani je još potrebno pod ključem *environment* definirati adresu Elasticsearch baze koja će biti izvor podataka.

5.1.1. *Dockerfile* za *wappalyzer-fetch* kontejner

Za razliku od ostalih kontjenera, *wappalyzer-fetch* ne koristi neku definiranu sliku, već u datoteci *Dockerfile* ima definiran sadržaj.

```
FROM node:alpine
RUN apk update

COPY . /app
COPY ./config/default.json /root/config/default.json
WORKDIR /app

RUN wget https://raw.githubusercontent.com/eficode/wait-
    for/master/wait-for
RUN chmod +x ./wait-for

RUN npm install

COPY entrypoint.sh /entrypoint.sh
ENTRYPOINT ["/entrypoint.sh"]

CMD ["crond", "-f", "-l", "2"]
```

Ispis 5.2: Sadržaj datoteke *Dockerfile*

Dockerfile iz Ispisa 5.2 započinje instrukcijom *FROM* kojom se definira bazna slika iz koje se gradi kontejner. Odabrana je slika *node:alpine* iz službenog repozitorija i sadrži Node.js baziran na Alpine Linux kontejneru.

Instrukcijom *RUN* pokreće se zadana naredba, *apk update* nadograđuje pakete koji se koriste na najnoviju verziju. Instrukcija *COPY* kopira datoteke iz lokalnog direktorija u kontejner. Ignoriraju se putanje navedene u datoteci *.dockerignore*, a to je direktorij *node_modules* u kojem se nalaze dodatni paketi za Node.js skriptu. Ti se dodaci instaliraju prilikom pokretanja servisa i zato je praksa ne dodavati ih u kontejner direktno.

Instrukcija *WORKDIR* postavlja radni direktorij unutar kontejnera. Nadalje, preuzima se skripta *wait-for* koja se koristi za sinkronizaciju kontejnera. Prilikom razvoja programa je ustanovljeno da se kontejner *wappalyzer-fetch* pokrene prije nego Elasticsearch baza bude postavljena i spremna primati zahtjeve. To bi uzrokovalo pad glavne skripte te se zato koristi *wait-for* alat koji odgađa početak rada programa dok Elasticsearch ne postane dostupan.

Zatim se pokreće *npm install* čime se instaliraju svi dodani paketi, navedeni u *package.json* datoteci pod ključem *dependencies*. Kasnije se kopira i pokreće datoteka

entrypoint.sh u kojoj se čeka s nastavkom izvođenja dok Elasticsearch ne bude dostupan.

Posljednja instrukcija je *CMD* kojom se definira zadana instrukcija za pokretanje kontejnera. Naredba *crond -f -l 2* pokreće proces cron koji izvršava zakazane naredbe.

5.2. Cron

Cron [19] je alat koji služi za zakazivanje poslova (engl. *job scheduling*) u definiranim vremenima. U *crontab* datoteci se definiraju intervali u kojima se treba izvesti proces. Svaki redak iz Ispisa 5.3 se sastoji od izraza za definiranje intervala i naredbe koja se izvodi.

Izraz za definiranje intervala se sastoji od pet brojeva. Umjesto broja može biti znak * koji predstavlja sve valjane brojeve za taj stupac.

- Prvi broj označava minutu u kojoj se treba izvršiti naredba (0-59)
- Drugi broj označava sat u kojem se treba izvršiti naredba (0-23)
- Treći broj označava dan u mjesecu kad se izvršava naredba (1-31)
- Četvrti broj označava mjesec u kojem se izvršava naredba (1-12)
- Peti broj označava dan u tjednu u kojem se izvršava naredba (0-6)

```
0 0 15 * * echo $(npm --prefix /app update wappalyzer)
10 0 15 * * echo $(npm --prefix /app update wappalyzer-core)
30 0 15 * * echo $(time node /app/app.js)
```

Ispis 5.3: Sadržaj datoteke *crontab*

Za primjer će se uzeti redak iz Ispisa 5.3:

```
0 0 15 * *
```

On označava da će se petnaestog dana u mjesecu u ponoć pokrenuti zadana naredba. To će se izvesti svaki mjesec jednom. Naredba koja se pokreće će nadograditi *wappalyzer* paket kako bi imali najnoviju bazu tehnologija. Deset minuta nakon (*10 0 15 * **) se nadograđuje paket *wappalyzer-core* koji sadrži skriptu za analizu. Konačno, svakog 15. dana u mjesecu u 00:30 se pokreće glavna skripta app.js koja analizira Web stranice dostupne u MongoDB bazi.

Svrha cron alata je da se dohvate najnovije promjene u bazi tehnologija prije nego se pokrene skripta. Osim toga, alat omogućuje redovno izvršavanje skripte bez potrebe za ručnim pokretanjem.

5.3. Node.js skripta

Prvi korak skripte je postavljanje vrijednosti konstante *startTimestamp*. U Ispisu 5.4 program provjerava je li korisnik pokrenuo skriptu s argumentom *--timestamp* i ako je onda postavlja *startTimestamp* u zadanu vrijednost. Inače *startTimestamp* dobije vrijednost vremenske oznake u trenutku pokretanja. To znači da nijedna Web stranica iz MongoDB baze neće biti preskočena tijekom Wappalyzer analize.

```
let pos = process.argv.findIndex(
    (arg) => arg === '--timestamp'
);

const startTimestamp = pos === -1
    ? Date.now() / 1000
    : Number(process.argv[pos + 1]);
```

Ispis 5.4: Isječak JavaScript koda koji određuje vrijednost varijable *startTimestamp*

Konstanta *startTimestamp* služi za filtriranje Web stranica iz MongoDB baze koje su analizirane nakon zadanog datuma i vremena. Taj se podatak skripti šalje pri pokretanju koristeći zastavicu –timestamp. Primjer:

```
node app.js --timestamp 1653858000
```

Takvo pozivanje skripte će analizirati samo one Web stranice koji su posljednji put bile analizirane prije 29.5.2022 21:00 GMT, ili one koje nisu uopće dosad bila poslane Wappalyzeru.

5.3.1. Postavljanje kategorija i tehnologija

Sljedeći korak je učitavanje tehnologija i kategorija iz baze. One se nalaze u paketu *wappalyzer*. Datoteka s kategorijama se nalazi na u datoteci

/app/node_modules/wappalyzer/categories.json

i koristeći ugrađenu funkciju *JSON.Parse()* se kategorije učitaju u objekt.

S obzirom na to da postoji velik broj tehnologija, one su raspodijeljene u datoteke ovisno o početnom slovu imena. Datoteke se nalaze u direktoriju

/app/node_modules/wappalyzer/technologies.

Imena datoteka su od *a.json* do *z.json*, uz posebnu datoteku *_.json* za tehnologije čije ime ne započinje slovom. Sadržaj svih datoteka s tehnologijama se učitava u JavaScript objekt.

5.3.2. Uspostava veze s MongoDB bazom podataka

Nakon postavljana kategorija i tehnologija za Wappalyzer, slijedi spajanje na bazu MongoDB kako je prikazano u Ispisu 5.5.

```
const username = config.get('mongo.username');
const password = config.get('mongo.password');

let url = 'mongodb://' + username + ':' + password +
    '@host.docker.internal/websecradar?authSource=admin';

await MongoClient.connect(url, ...
```

Ispis 5.5: Povezivanje s MongoDB bazom podataka

U Ispisu 5.5 se prvo dohvati korisničko ime i zaporka iz *config/default.json* datoteke.

```
{
        "mongo": {
                "username": "foo",
                "password": "bar"
        }
}
```

Ispis 5.6: Primjer ispravne *default.json* datoteke

Ispis 5.6 prikazuje sadržaj datoteke *default.json.example* datoteke. Vrijednosti ključeva *username* i *password* se trebaju zamijeniti pravim korisničkim imenom i zaporkom unutar datoteke *default.json*.

Zatim se stvara URL za spajanje na bazu *websecradar*. Konačno se u Ispisu 5.5 funckijom *connect* nad objektom MongoClient pokušava uspostaviti veza s bazom.

5.3.3. Dohvat adresa Web stranica iz kolekcije *urls*

Nakon uspješnog povezivanja s MongoDB bazom, slijedi dohvaćanje dokumenata iz kolekcije *urls*. Dokumenti se dohvaćaju u hrpama (engl. *batch*) od 100 dokumenata.

Svaki dokument sadrži podatke o jednoj Web stranici.

Upit iz Ispisa 5.7 koristi uvjet *$or* kojim definira da se odaberu samo oni dokumenti koji nemaju postavljeno polje *wappalyzer_processing* ili je vrijednost polja manja od varijable *startTimestamp*. Uvjet *$lt* (skraćeno od *less than*) predstavlja operaciju usporedbe manje od. Kao što je objašnjeno u poglavlju *Node.js skripta*, to osigurava da se Wappalyzeru šalju samo one Web stranice koje su posljednji put poslane na analizu prije zadanog datuma. Polje *wappalyzer_processing* se ažurira svaki put kad se Web stranica analizira. Opcijom *projection* se definiraju polja iz dokumenta koja se žele dohvatiti.

```
const domains = await mongoDb.collection('urls')
  .find(
    { $or : [
        { 'wappalyzer_processing': {$lt: startTimestamp}},
        { 'wappalyzer_processing': { $exists: false }}
      ]
    },
    { projection: { url: 1, _id: 0 } }
  )
  .limit(100)
  .toArray();
```

Ispis 5.7: Isječak koda koji dohvaća dokumente iz *urls* kolekcije

Slika 5.1 prikazuje jedan dokument it kolekcije *urls*. S obzirom na to da je *url* jedino polje iz kolekcije *urls* koje se koristi u skripti, u opciji *projection* označujemo da se dohvati to polje. Eksplicitno označujemo da se ne dohvaća polje *_id* koje se automatski inače uzima. Takav selektivan odabir polja smanjuje promet između skripte i MongoDB baze.

Slika 5.1: Primjer dokumenta iz *urls* kolekcije

5.3.4. Dohvat sadržaja Web stranica

Dobiveno polje *url* nam služi kao ključ za dohvat ostalih podataka o Web stranicama iz *websecradar* baze. Prvi korak je dohvatiti odgovarajući dokument iz kolekcije *crawled_data_urls_v0*.

Slika 5.2: Primjer dokumenta iz kolekcije *crawled_data_urls_v0*

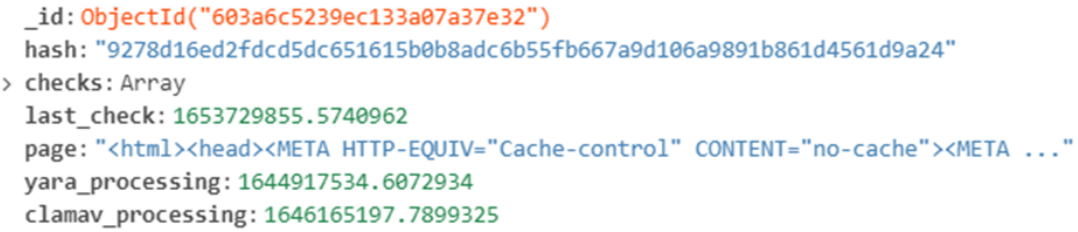
Slika 5.2 prikazuje jedan dokument iz kolekcije *crawled_data_urls_v0*. U nizu *checks* iz Slike 5.2 se nalaze osnovne informacije o dohvaćanjima Web stranice:

- Vrijeme (vremenska oznaka)
- Zaglavlja HTTP odgovora
- HTTP statusni kod

Slika 5.3: Zapis unutar niza *checks*

– Sažetak (engl. *hash*) HTML sadržaja stranice

Zaglavlja iz Slike 5.3 se spremaju u varijablu za kasniju uporabu, a polje *hash* se koristi za dohvat HTML sadržaja iz kolekcije *crawled_data_pages_v0*



Slika 5.4: Primjer dokumenta iz kolekcije *crawled_data_pages_v0*

Polje *page* iz Slike 5.4 sadrži HTML kod stranice i ono se će se koristiti za Wappalyzer analizu. U nizu *checks* iz Slike 5.4 se nalazi popis JavaScript i CSS datoteka koje stranice referencira. Sažeci tih datoteka koriste se za dohvat sadržaja. Kasnije će se zajedno uz HTML slati na Wappalyzer analizu.

Neki dokumenti u kolekciji *crawled_data_pages_v0* nemaju podatke o referenciranim JavaScript datotekama. Zato se i iz HTML koda izvlače ti podaci, kao što je prikazano u Ispisu 4.5 i Ispisu 4.6. Za to se koristi paket *node-html-parser*.

5.3.5. Prilagođavanje podataka iz MongoDB baze

U poglavlju *Meta podaci* je opisano kako je potrebno meta podatke iz HTML sadržaja stranice prilagoditi kako bi ih Wappalyzer interpretirati. To se obavlja u funkciji *extractMeta*.

Prvi korak funckije iz Ispisa 5.8 je dohvaćanje svih *meta* elemenata. Po njima se iterira te ako imaju atribute *name* i *content* onda se pohranjuju u objekt koji se kasnije šalje Wappalyzeru. Zbog načina kako Wappalyzer upravlja podacima iz baze, potrebno je poslati dvije verzije meta podataka. Jedna je originalna, a drugoj se sadržaj iz atributa *name* pretvara u mala slova (engl. *lowercase*). Primijećeno je tijekom testiranja da Wappalyzer ne detektira verzije WordPressa na određenim stranicama. Nakon istraživanja izvornog koda Wappalyzera je utvrđeno da se niz znakova "Generator" i "generator" nisu podudarali i zbog toga nisu dolazili željeni rezultati.

```
const metaTags = HTMLParser.parse(html).querySelectorAll('meta');

for(const metaTag of metaTags) {
    const name = metaTag.getAttribute('name');
    const content = metaTag.getAttribute('content');

    meta[name].push(content);
    meta[name.toLowerCase()].push(content);
}
```

Ispis 5.8: Simplificirana verzija funkcije za parsiranje meta podataka

Isti proces se obavlja nad podacima o zaglavljima HTTP odgovora. Dupliciraju se ključevi kako bi imali i verziju s malim slovima. Osim toga, Wappalyzer zahtjeva da se sve vrijednosti zaglavlja nalaze unutar niza.

Izvorno zaglavlje

```
{
    Server: "Apache"
}
```

nakon transformacije poprima sljedeći oblik:

```
{
    Server: ["Apache"],
    server: ["Apache"],
}
```

5.3.6. Pokretanje Wappalyzer analize

Svi dohvaćeni i pripremljeni podaci se šalju Wappalyzeru na analizu. Wappalyzer vraća popis detektiranih tehnologija. Sljedeća zadaća je prepoznati one tehnologije koje spadaju u sustave za upravljanje sadržajem.

U varijabli *results* iz Ispisa 5.9 se nalaze sve detektirane tehnologije. Svaka od njih je dio jedne ili više kategorija. Po tim kategorijama se iterira i uspoređuje ih s konstantom *CMS_CATEGORY_ID*, koja sadrži identifikator kategorije sustava za upravljanje sadržajem. Za sve uspješne provjere se poziva funkcija *addToElasticsearch* koja će napraviti novi unos u Elasticsearch bazu podataka.

```
for (const technology of results) {
   for (const category of technology.categories) {
      if (category.id === CMS_CATEGORY_ID) {
         await addToElasticsearch(
            url,
            technology.name,
            technology.version,
            technology.confidence,
            currentTimestamp,
            hash
         ).catch(console.log);
      }
   }
}
```

Ispis 5.9: Prepoznavanje tehnologija koje spadaju u CMS kategoriju

5.3.7. Unos rezultata u Elasticsearch

Za pohranu rezultata se koristi indeks *websecradar-detection-wappalyzer*. Naziv prati konvenciju imenovanja Elasticsearch indeksa na poslužitelju ZEMRIS-a.

Ispis 5.10 prikazuje podatke koji se šalju u Elasticsearch. Prva četiri polja su napravljena kako bi se podudarala s ostalim servisima na poslužitelju na Zavodu. Ti zapisi se ne šalju odmah čim se detektira CMS, već se nakupljaju i šalju u grupama (engl. *bulk*) kako bi bilo manje opterećenje na Elasticsearch bazi.

```
{
        timestamp: timestamp,
        page_hash: page_hash,
        match_rule: slugify(cms_name),
        rule_hash: slugify(cms_name + " " + cms_version),

        main_url: url,
        cms_name: cms_name,
        cms_version: cms_version,
        cms_version_defined: !(cms_version === undefined),
        cms_confidence: confidence
}
```

Ispis 5.10: Objekt koji se šalje u Elasticsearch

timestamp – vremenska oznaka kad je obavljena analiza

page_hash – sažetak stranice iz MongoDB kolekcije

match_rule – slug vrijednost imena sustava za upravljanje sadržajem

rule_hash – kombinacija naziva i verzija sustava za upravljanje sadržajem

main_url – URL Web stranice

cms_name – naziv sustava za upravljanje sadržajem

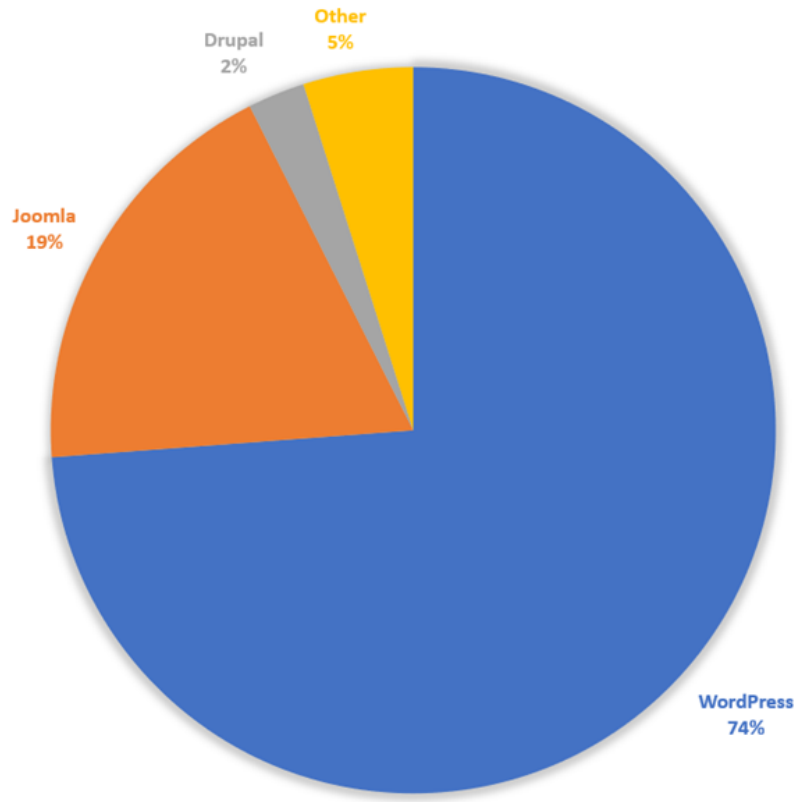
cms_version – verzija sustava za upravljanje sadržajem

cms_version_defined – *boolean* vrijednost, istinita ako je detektirana verzija

cms_confidence – mjerilo Wappalyzera za sigurnost detekcije

6. Rezultati

Ukupan broj analiziranih stranica je 267 941, a uključuje sve stranice dostupne u kolekciji *urls* kad je servis pokrenut. Bilo je potrebno 19 sati da servis prođe kroz cijelu bazu podataka. Wappalyzer je detektirao 36 933 (13.78%) Web stranica koje koriste sustav za upravljanje sadržajem. Wappalyzer je uspio detektirati verziju sustava za upravljanje sadržajem za 23 479 (63.57%) od tih 39 933 Web stranica.

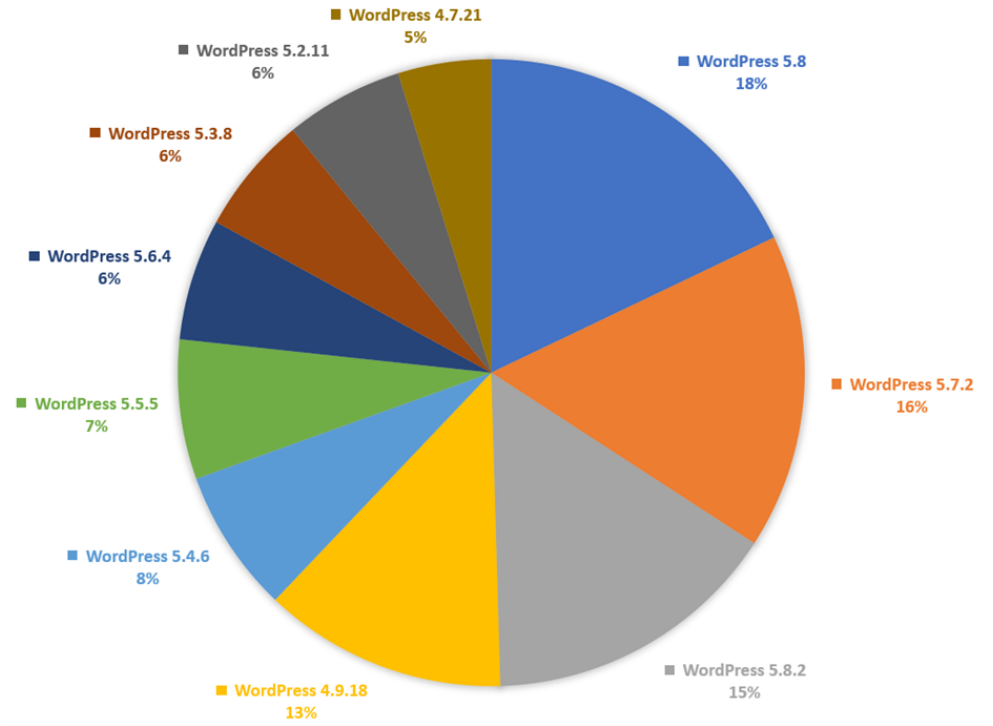


Slika 6.1: Udio pojedinih sustava za upravljanje sadržajem

U poglavlju *WordPress* je rečeno da WordPress dominira tržištem sustava za upravljanje sadržajem. Rezultati Wappalyzer analize nad podacima iz MongoDB baze pokazuju isto. Na Slici 6.1 se vidi da od svih stranica koje koriste neki CMS, 74% ih

koristi WordPress. Drugo mjesto zauzima Joomla s 19%, a treće Drupal sa 2%. Svaki od preostalih sustava za upravljanje sadržajem se pojavljuje u manje od 1% stranica. Ukupno je detektirano 58 različitih sustava za upravljanje sadržajem. Najpopularnijih deset su WordPress, Joomla, Drupal, Wix, Craft CMS, TYPO3 CMS, DNN, Weebly, Adobe Experience Manager i eZ Publish.

Detaljnijim uvidom u rezultate Wappalyzera za WordPress mogu se izdvojiti najpopularnije verzije ovog sustava za upravljanje sadržajem.



Slika 6.2: Udio verzija WordPressa

Na Slici 6.2 je prikazano deset najkorištenijih verzija WordPressa na dan kad je pokrenuta analiza, 19. travnja 2022. Od ukupno 27 263 WordPress stranica, 10,000 ih je uključeno u statistiku sa Slike 6.2, a 5 902 WordPress stranica nema detektiranu verziju. Osim prikazanih deset verzija, Wappalyzer ih je prepoznao još 310 koje zbog malog udjela nisu uključene u graf na Slici 6.2.

Najnovija *major* verzija WordPressa, 5.9, objavljena je 25. siječnja 2022. Četiri mjeseca kasnije kad je pokrenuta Wappalyzer analiza, u MongoDB bazi se nalazilo 553 Web stranica s verzijom WordPressa 5.9 ili novijom. To je 2.028% od svih WordPress stranica. Verziju 5.9.3, koja je objavljena 5. travnja 2022., je koristila 341 Web stranica u trenutku analize. U obzir treba i uzeti mogućnost da neke od Web stranica iz MongoDB baze su sadržavale starije podatke.

Za svrhe ovog rada koji se primarno fokusirao na tehničku implementaciju sustava je obavljena kraća analiza rezultata kao uvid u ono što je moguće postići s rezultatima. U obzir su uzeti općenito sustavi za upravljanje sadržajem i WordPress kao najpoznatiji sustav za upravljanje sadržajem. Detaljnija analiza rezultata nije uključena u ovaj rad.

7. Zaključak

Napravljen je programski sustav koji analizira Web stranice iz baze Web stranica na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave Fakulteta elektrotehnike i računarstva. Glavni cilj je detektirati sustave za upravljanje sadržajem i verzije koje se koriste. Sustav se pokreće koristeći Docker, skripta koja analizira Web stranice je napisana koristeći Node.js, a rezultati se spremaju u Elasticsearch bazu podataka.

Sljedeći korak je nadogradnja sustava kako bi uspoređivao detektirane verzije s podacima o ranjivim verzijama sustava za upravljanje sadržajem. Osim toga, nad već dobivenim rezultatima analize je moguće dobiti razne statistike o popularnosti sustava za upravljanje sadržajem. Periodičkom analizom Web stranica mogu se vidjeti navike Web stranica što se tiče nadogradnje CMS-ova na novije verzije.

8. Literatura

[1] M. Meike, J. Sametinger, and A. Wiesauer, "Security in open source web content management systems," *IEEE Security & Privacy*, vol. 7, no. 4, pp. 44–51, 2009.

[2] J. Cabot, "WordPress: A Content Management System to Democratize Publishing," *IEEE Software*, vol. 35, no. 3, pp. 89–92, 2018.

[3] "Usage statistics and market share of WordPress," https://w3techs.com/technologies/details/cm-wordpress, (pristupljeno 22. svibanj 2022.).

[4] D. Balaban, "The endless story of WordPress plugins security issues," https://www.itworldcanada.com/blog/the-endless-story-of-wordpress-plugins-security-issues/431784, (pristupljeno 22. svibanj 2022.).

[5] R. Abela, "Statistics Show Why WordPress is a Popular Hacker Target," https://www.wpwhitesecurity.com/statistics-70-percent-wordpress-installations-vulnerable/, (pristupljeno 22. svibanj 2022.).

[6] "Brute Force Attacks," https://wordpress.org/support/article/brute-force-attacks//, (pristupljeno 22. svibanj 2022.).

[7] J. Clarke, *SQL injection attacks and defense*. Elsevier, 2009.

[8] K. Wright, "SQL injection: A Guide for WordPress Users," https://ithemes.com/blog/sql-injection-wordpress/, (pristupljeno 26. svibanj 2022.).

[9] "How to prevent SQL Injection vulnerabilities: How Prepared Statements Work," https://www.hackedu.com/blog/how-to-prevent-sql-injection-vulnerabilities-how-prepared-statements-work/, (pristupljeno 26. svibanj 2022.).

[10] “WordPress: CVE security vulnerabilities, versions and detailed reports,” https://www.cvedetails.com/product/4096/Wordpress-Wordpress.html?vendor_id=2337, (pristupljeno 26. svibanj 2022.).

[11] “CMS attacks on the rise,” https://services.global.ntt/en-us/insights/blog/cms-attacks-on-the-rise, (pristupljeno 19. svibanj 2022.).

[12] “About Node.js,” https://nodejs.org/en/about/, (pristupljeno 16. svibanj 2022.).

[13] “Introduction to MongoDB,” https://www.mongodb.com/docs/manual/introduction/, (pristupljeno 16. svibanj 2022.).

[14] “What is Elasticsearch?” https://www.elastic.co/what-is/elasticsearch, (pristupljeno 16. svibanj 2022.).

[15] “What is Kibana?” https://www.elastic.co/what-is/kibana, (pristupljeno 16. svibanj 2022.).

[16] “Docker overview,” https://docs.docker.com/get-started/overview/, (pristupljeno 16. svibanj 2022.).

[17] “Overview of Docker Compose,” https://docs.docker.com/compose/, (pristupljeno 16. svibanj 2022.).

[18] “Find out what websites are built with - Wappalyzer,” https://www.wappalyzer.com/, (pristupljeno 26. svibanj 2022.).

[19] Wikipedia contributors, “Cron — Wikipedia, the free encyclopedia,” (pristupljeno 27. svibanj 2022). [Online]. Available: https://en.wikipedia.org/w/index.php?title=Cron&oldid=1091715941

Detekcija korištenih Web tehnologija upotrebom alata Wappalyzer

Sažetak

Sustavi za upravljanje sadržajem (*engl. content management system, CMS*) su popularna rješenja za stvaranje Web stranica. Većina sustava za upravljanje sadržajem je otvorenog koda koji je dostupan napadačima. Zbog toga su i jedni od najpopularnijih meta na Internetu. Vrlo je bitno koristiti verzije sustava za upravljanje sadržajem s najnovijim sigurnosnim ažuriranjima.

Ovaj rad i program su napravljeni sa svrhom detekcije Web stranica *hr* domene koje koriste neki sustav za upravljanje sadržajem. Cilj je detektirati verziju sustava za upravljanje sadržajem na Web stranicama. Ta informacije se može kasnije iskoristiti kako bi se saznalo je li riječ o verziji koja je ranjiva.

Wappalyzer je alat specijaliziran za identifikaciju tehnologija koje Web stranice koriste. Korištenjem alata Wappalyzer je napravljen sustav koji periodički analizira Web stranice iz baze Web stranica *hr* domene i uspješne detekcije sustava za upravljanje sadržajem zapisuje u Elasticsearch. Cijeli sustav se pokreće u Docker kontejnerima.

Analiza je provedena nad 267 941 Web stranica i uspješnost detekcije verzije CMS-a na stranicama koje ih koriste je 63.57%. Među sustavima za upravljanje sadržajem dominira WordPress po popularnosti. Rezultati analize dostupni u Elasticsearch bazi mogu se iskoristiti za detaljnije statistike o sustavima za upravljanje sadržajem u *hr* domeni.

Ključne riječi: detekcija Web tehnologija, Wappalyzer, sustav za upravljanje sadržajem, CMS, ranjivosti

Detection of used Web technologies with Wappalyzer tool

Abstract

Content management systems (CMS) are popular solution when it comes to creating Websites. Most of them are open source and their code is available to attackers. That makes them one of the most popular targets on the Internet. It is important to use versions of CMS which include latest security patches.

This program is created with a purpose of detecting Websites in *hr* domain which use CMS. The goal is to detect which version of CMS is used. That information can be later used to see if Website is using vulnerable version of CMS.

Wappalyzer is a tool specialized for identification of technologies which Websites use. Using Wappalyzer, system is built which periodically analyses Websites from database of Websites from *hr* domain. Successful CMS detections are stored in Elasticsearch. System is running in Docker containers.

The analysis was performed on 267 941 Websites. Success rate of version detection on Websites which use CMS is 64.57%. Among content management systems, WordPress is the most popular one. Results of analysis are stored in Elasticsearch and can be used for detailed statistics about content management systems used in Websites in *hr* domain.

Keywords: detection of Web technologies, Wappalyzer, content management system, CMS, vulnerabilities