01

UPORABNA INFORMATIKA

2025 ‹ ŠTEVILKA 1 ‹ LETNIK ????? ‹ ISSN 1318-1882

U P O R A B N A
I N F O R M A T I K A
2026 ŠTEVILKA 1 JAN/FEB/MAR LETNIK XXXIV ISSN 1318-1882

Znanstveni prispevki

Strokovni prispevki

Informacije

In memoriam

2026 ŠTEVILKA 1 JAN/FEB/MAR LETNIK XXXIV ISSN 1318-1882

Ustanovitelj in izdajatelj
Slovensko društvo INFORMATIKA
Litostrojska cesta 54, 1000 Ljubljana

Predstavnik
Slavko Žitnik

Odgovorni urednik
Mirjana Kljajić Borštnar

Prelom in tisk
Boex DTP, d. o. o., Ljubljana

Naklada
110 izvodov

Naslov uredništva
Slovensko društvo INFORMATIKA
Uredništvo revije Uporabna informatika
Litostrojska cesta 54, 1000 Ljubljana
www.uporabna-informatika.si

Revija izhaja četrtletno. Cena posamezne številke je 20,00 EUR. Letna naročnina za podjetja 85,00 EUR, za vsak nadaljnji izvod 60,00 EUR, za posameznike 35,00 EUR, za študente in seniorje 15,00 EUR. V ceno je vključen DDV.

Revija Uporabna informatika je od številke 4/VII vključena v mednarodno bazo INSPEC.

Revija Uporabna informatika je pod zaporedno številko 666 vpisana v razvid medijev, ki ga vodi Ministrstvo za kulturo RS.

Revija Uporabna informatika je vključena v Digitalno knjižnico Slovenije (dLib.si).

Izid publikacije je finančno podprla Javna agencija za znanstvenoraziskovalno in inovacijsko dejavnost Republike Slovenije.

Vabilo avtorjem

V reviji Uporabna informatika objavljamo kakovostne izvirne prispevke domačih in tujih avtorjev z najširšega področja informatike, ki se nanašajo tako na poslovanje podjetij, javno upravo, družbo in posameznika. Prispevki so lahko znanstvene, strokovne ali informativne narave, še posebno spodbujamo objavo interdisciplinarnih prispevkov. Zato vabimo avtorje, da prispevke, ki ustrezajo omenjenim usmeritvam, pošljejo uredništvu revije po elektronski pošti na naslov ui@drustvo-informatika.si.

Avtorje prosimo, da pri pripravi prispevka upoštevajo navodila, ki so objavljena na naslovu http://www.uporabna-informatika.si.

Za kakovost prispevkov skrbi mednarodni uredniški odbor. Prispevki so anonimno recenzirani, o objavi pa na podlagi recenzij samostojno odloča uredniški odbor. Recenzenti lahko zahtevajo, da avtorji besedilo spremenijo v skladu s priporočili in da popravljeni prispevek ponovno prejmejo v pregled. Sprejeti prispevki so pred izidom revije objavljeni na spletni strani revije (predobjava), še prej pa končno verzijo prispevka avtorji dobijo v pregled in potrditev. Uredništvo lahko še pred recenzijo zavrne objavo prispevka, če njegova vsebina ne ustreza vsebinski usmeritvi revije ali če prispevek ne ustreza kriterijem za objavo v reviji.

Pred objavo prispevka mora avtor podpisati izjavo o avtorstvu, s katero potrjuje originalnost prispevka in dovoljuje prenos materialnih avtorskih pravic. Avtorji prejmejo enoletno naročnino na revijo Uporabna informatika, ki vključuje avtorski izvod revije in še nadaljnje tri zaporedne številke. S svojim prispevkom v reviji Uporabna informatika boste pomagali k širjenju znanja na področju informatike. Želimo si čim več prispevkov z raznoliko in zanimivo tematiko in se jih že vnaprej veselimo

Uredništvo revije

Navodila avtorjem člankov

Članke objavljamo praviloma v slovenščini, članke tujih avtorjev pa v angleščini. Besedilo naj bo jezikovno skrbno pripravljeno. Priporočamo zmernost pri uporabi tujk in, kjer je mogoče, njihovo zamenjavo s slovenskimi izrazi. V pomoč pri iskanju slovenskih ustreznic priporočamo uporabo spletnega terminološkega slovarja Slovenskega društva Informatika, Islovar (www.islovar.org).

Znanstveni prispevek naj obsega največ 40.000 znakov, kratki znanstveni prispevek do 10.000 znakov, strokovni članki do 30.000 znakov, obvestila in poročila pa do 8.000 znakov.

Prispevek naj bo predložen v urejevalniku besedil Word (*.doc ali *.docx) v enojnem razmaku, brez posebnih znakov ali poudarjenih črk. Za ločilom na koncu stavka napravite samo en presledek, pri odstavkih ne uporabljajte zamika.

Naslovu prispevka naj sledi polno ime vsakega avtorja, ustanova, v kateri je zaposlen, naslov in elektronski naslov. Sledi naj povzetek v slovenščini v obsegu 8 do 10 vrstic in seznam od 5 do 8 ključnih besed, ki najbolje opredeljujejo vsebinski okvir prispevka. Sledi naj prevod naslova povzetka in ključnih besed v angleškem jeziku. V primeru, da oddajate prispevek v angleškem jeziku, velja obratno. Razdelki naj bodo naslovljeni in oštevilčeni z arabskimi številkami.

Slike in tabele vključite v besedilo. Opremite jih z naslovom in oštevilčite z arabskimi številkami. Na vsako sliko in tabelo se morate v besedilu prispevka sklicevati in jo pojasniti. Če v prispevku uporabljate slike ali tabele drugih avtorjev, navedite vir pod sliko oz. tabelo. Revijo tiskamo v črno-beli tehniki, zato barvne slike ali fotografije kot original niso primerne. Slikam zaslonov se v prispevku izogibajte, razen če so nujno potrebne za razumevanje besedila. Slike, grafikoni, organizacijske sheme ipd. naj imajo belo podlago. Enačbe oštevilčite v oklepajih desno od enačbe.

V besedilu se sklicujte na navedeno literaturo skladno s pravili sistema IEEE navajanja bibliografskih referenc, v besedilu to pomeni zaporedna številka navajenega vira v oglatem oklepaju (npr. [1]). Na koncu prispevka navedite samo v prispevku uporabljeno literaturo in vire v enotnem seznamu, urejeno po zaporedni številki vira, prav tako v skladu s pravili IEEE. Več o sistemu IEEE, katerega uporabo omogoča tudi urejevalnik besedil Word 2007, najdete na strani https://owl.purdue.edu/owl/research_and_citation/ieee_style/ieee_general_format.html.

Prispevku dodajte kratek življenjepis vsakega avtorja v obsegu do 8 vrstic, v katerem poudarite predvsem strokovne dosežke.

Sistematična analiza napada POODLE na SSL 3.0 z AES-CBC in ocena protiukrepov

Anja Klančar, Matevž pesek
Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Večna pot 113, 1000 Ljubljana
ak01265@student.uni-lj.si, matevz.pesek@fri.uni-lj.si

Izvleček

V pričujočem članku obravnavamo kriptografski napad POODLE (Padding Oracle On Downgraded Legacy Encryption), ki izkorišča ranljivost protokola SSL 3.0 pri uporabi šifriranja AES v načinu CBC skupaj s pristopom overi-nato-šifriraj. Najprej predstavimo teoretično ozadje napada, vključno z vplivom podlage, veriženja blokov in razločevanja med različnimi tipi napak pri dešifriranju. Nato prikažemo, kako lahko napadalec z opazovanjem odzivov strežnika postopoma razkriva podatke brez poznavanja skrivnega ključa. V praktičnem delu opišemo implementacijo AES-CBC v programskem jeziku Java, izvedbo napada POODLE ter analizo pogojev, ki omogočajo njegovo uspešnost. Posebej se osredotočimo na dve obrambni strategiji: poenotenje odzivov ob napaki, pri katerem strežnik ne razkriva vrste napake, in uporabo pristopa šifriraj-nato-overi, ki prepreči napad že pred dešifriranjem podatkov. Rezultati pokažejo, da je POODLE posledica neustrezne kombinacije kriptografskih mehanizmov in da je varnost takšnih sistemov močno odvisna tudi od pravilne implementacije protiukrepov.

Ključne besede: kriptografija, kriptografski napad, POODLE, SSL

SYSTEMATIC ANALYSIS OF THE POODLE ATTACK ON SSL 3.0 WITH AES-CBC AND EVALUATION OF COUNTERMEASURES

Abstract

In this paper, we examine the POODLE (Padding Oracle On Downgraded Legacy Encryption) cryptographic attack, which exploits a vulnerability in the SSL 3.0 protocol when AES encryption in CBC mode is used together with the MAC-then-encrypt approach. We first present the theoretical background of the attack, including the role of padding, block chaining, and the distinction between different types of decryption errors. We then show how an attacker can gradually recover data by observing server responses without knowing the secret key. In the practical part, we describe an implementation of AES-CBC in the Java programming language, the execution of the POODLE attack, and an analysis of the conditions that enable its success. Particular attention is given to two defense strategies: unifying error responses so that the server does not reveal the type of error, and using the encrypt-then-MAC approach, which prevents the attack before any data is decrypted. The results show that POODLE is a consequence of an inappropriate combination of cryptographic mechanisms and that the security of such systems also depends strongly on the correct implementation of countermeasures.

Keywords: cryptography, cryptographic attack, POODLE, SSL

1 UVOD

Kriptografija je temeljni gradnik sodobnih informacijskih sistemov, saj zagotavlja varno komunikacijo, zaščito podatkov in ohranja zaupanje v digitalnih okoljih. Poleg varovanja zaupnosti omogoča tudi preverjanje pristnosti, celovitosti in nezanikljivosti informacij, kar je ključno pri prenosu občutljivih podatkov v kritičnih aplikacijah, kot so bančništvo, elektronsko poslovanje in varna spletna komunikacija.
Cilji kriptografije so običajno opredeljeni v štirih kategorijah: zaupnost, celovitost, avtentikacija in nezanikljivost. Zaupnost zagotavlja, da imajo dostop do podatkov zgolj avtorizirane osebe, kar se doseže s šifriranjem. Celovitost pomeni, da se podatki med prenosom ne spremenijo, kar preverimo s pomočjo zgoščevalnih funkcij. Avtentikacija zagotavlja, da je pošiljatelj sporočila resničen, kar se v praksi izvaja s kodo za preverjanje sporočila (MAC), ki obenem prispeva k ohranjanju celovitosti. Nezanikljivost pa zagotavlja, da pošiljatelj ne more zanikati svoje udeležbe pri komunikaciji, kar se dosega z digitalnimi podpisi [1].

Med protokoli, ki udejanjajo te varnostne lastnosti, je ključnega pomena protokol SSL (Secure Socket Layer), namenjen vzpostavitvi varnega prenosa informacij prek omrežij. Protokol je sestavljen iz dveh slojev: *Record* plasti, ki skrbi za zaupnost, avtentičnost in zaščito pred napadi s ponovitvijo, ter *Handshake* protokola, ki omogoča izmenjavo ključev, inicializacijo komunikacije in uskladitev kriptografskega stanja med odjemalcem in strežnikom [2].

Kasneje zastarela najnaprednejša različica SSL protokola je bila različica SSL 3.0, ki je nadomestila različico SSL 2.0 zaradi več varnostnih pomanjkljivosti, kot so uporaba 40-bitnih ključev za avtentikacijo v nekaterih načinih, šibek mehanizem MAC in pomanjkljiva avtentikacija polja dolžine podloge, ki je omogočala manipulacijo z bajti na koncu sporočila. Kot naslednik SSL 3.0 je bil vzpostavljen protokol TLS 1.0 (Transport Layer Security), ki je uvedel pomembne izboljšave, med drugim uporabo varnejšega HMAC. Vendar pa ostaja ranljivost v obliki t.i. "downgrade attack", kjer lahko napadalec prisili odjemalca in strežnik, da komunikacijo vzpostavita prek starejše, manj varne različice SSL 3.0, kar predstavlja osnovo za izvedbo POODLE napada [3]. Najnovejša različica protokola TLS je različica TLS 1.3, ki je danes najbolj priporočljiva za uporabo. V primerjavi s predhodnimi različicami je prinesel veliko izboljšav, kot so izboljšana varnost, hitrejše delovanje in večjo fleksibilnost [4].

V tem članku je opisan kriptografski napad POODLE (Padding Oracle On Downgraded Legacy Encryption), ki temelji na ranljivosti protokola SSL 3.0 (Secure Socket Layer). Protokol uporablja AES (Advanced Encryption Standard) šifriranje v CBC (Cipher Block Chaining) načinu, kar samo po sebi ni nevarno, problem pa postane, če se ta način uporablja skupaj z načinom overi-nato-šifriraj. Problem z overi-nato-šifriraj je, da mora strežnik sporočilo najprej dešifrirati in šele nato lahko preveri overitveno značko. Zaradi takšnega delovanja nam strežnik v primeru napake pri dešifriranju pove tip napake, do katere je prišlo. Za napad sta pomembna dva tipa napak: napaka podloge in napaka značke. Nadaljnje predstavimo dve rešitvi preprečitve napada. Prva predstavljena rešitev je da ne glede na tip napake strežnik vrne napako značke, kar mora biti v praksi pravilno implementirano, saj drugače odpre vrata za časovni napad — torej, če je v resnici napaka podloge, je napaka hitreje vrnjena, kot pa če je napaka značke. V drugi rešitvi pa je predstavljena implementacija z načinom šifriraj-nato-overi, pri katerem se mora najprej preveriti značka, preden se dešifriranje sploh začne in bi se zato spreminjanje tajnopisa pravočasno ugotovilo, preden bi bil napad lahko izveden.

2 PREGLED PODROČJA

Zaradi razvoja informacijske in komunikacijske tehnologije je le ta prisotna povsod — v podjetjih, šolah, državnih inštitucijah, po domovih in drugi infrastrukturi. Zaradi njene razširjenosti so vse bolj aktualni napadi na infrastrukturo in podatke, imenovani kibernetski napadi, zaščita pred temi napadi pa se imenuje kibernetska varnost. Kibernetski napadi se izvajajo z namenom pridobivanja finančne koristi, vohunjenja, posebne oblike vojskovanja in druge [5].

Kibernetske napade lahko ločimo tudi med ciljno usmerjenimi napadi (angl. targeted attacks) in neciljnimi napadi (angl. untargeted attacks). Pri ciljno usmerjenih napadih si napadalec izbere specifičen cilj (specifično podjetje, sistem ipd.) in ga napade. Takšni napadi so ponavadi večja grožnja kot neciljni napadi, saj je napad narejen specifično za to tarčo in njene ranljivosti. Pri neciljnih napadih gre za napade, ki poskušajo ciljati čim več naprav ali uporabnikov hkrati. Za distribucijo večinokrat uporabljajo inter-

net. Primera takšnih napadov sta na primer ribarjenje (angl. phishing) in izsiljevalska programska oprema (angl. ransomware). [6]

Uspešni napadi imajo lahko katastrofalne posledice. Na državni ravni so to lahko na primer grožnja nacionalni varnosti, škoda gospodarskim in političnim odnosom države in škoda nacionalnemu gospodarstvu [7]. Kibernetske napade, ki so povezani s tehnologijo šifriranja, opredelimo kot kriptografske napade [8]. Kriptografske napade lahko razdelimo na več vrst:

Napadi na razpoložljivost (angl. Avalibility attacks) so napadi, katerih cilj je onemogočiti dostop do podatkov. Najpogostejši izmed njih je napad DoS (Denial-of-service).

Napadi na celovitost (angl. Integrity attacks) so napadi s ciljem uničenja podatkov. Te vrste napadov je veliko težje zaznavati kot napade razpoložljivosti, saj so velikokrat bolj prefinjeni.

Napadi na zaupnost (angl. Confidentiality attacks) so napadi, ki poskušajo ponarediti ali ukrasti zaupne informacije. Ti tipi napadov pogosto vključujejo tudi uporabo drugih dveh vrst napadov. [9]

POODLE napad spada v kategorijo napadov na celovitost, saj omogoča branje zaupnih informacij, do katerih napadalec ne bi smel imeti dostopa. Obstaja več kriptografskih napadov, ki so v nekaterih pogledih podobni POODLE, ki ga naslavljamo v pričujočem članku. Dva od teh sta CRIME-napad in Bleichenbacherjev napad. CRIME izkorišča ranljivost istega protokola kot POODLE, torej SSL 3.0, Bleichenbacherjev napad pa spada v isto skupino napadov, torej Padding Oracle attacks, za katere je značilno, da za napad izkoriščajo podlogo zakriptiranih podatkov.

2.1 Crime

Ta napad izkorišča lastnosti kompresijske metode DEFLATE, ki je uporabljena v TLS do verzije 1.2 in v SSL 3.0. Ta metoda med drugim podatke skrči, tako da zamenja instance enakih nizov s kazalcem, ki kaže iz prvega enakega niza na drugega in z dolžino niza. Po kompresiji podatki niso vidni, je pa vidna velikost skompresirane poizvedbe. To pomeni, da bo končna poizvedba manjša, če je veliko nizov enakih, kot pa če so si vsi med sabo različni.
Odjemalčeve poizvedbe so lahko videti na primer takole:

```
POST / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100
101 Firefox/14.0.1
Cookie: secretcookie=7xc89f94wa96fd7cb4cb0031ba249ca2
Accept-Language: en-US,en;q=0.8
```

Napadalec začne s poizvedbo, v kateri nastavi piškotek na 0 in opazuje velikost skompresirane poizvedbe:

```
POST /secretcookie=0 HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:14.0) Gecko/20100
101 Firefox/14.0.1
```

Piškotek povečuje za 1, dokler ne opazi, da je velikost skompresirane poizvedbe manjša kot prej. Ko se to zgodi, pomeni, da je bil prvi bajt piškotka pravilno ugotovljen, saj je bil ponavljajoči niz daljši. V našem primeru bi to bil bajt z vrednostjo 7. Torej ko je bil bajt nastavljen na 0, je bil ponavljajoči niz samo: secretcookie=, ko pa smo bajt nastavili na 7, pa je ponavljajoči niz postal: secretcookie=7 in je bila zato velikost končne skompresirane poizvedbe manjša. [10] [11]

2.2 Bleichenbacherjev napad

Bleichenbacherjev napad, prav tako kot POODLE, izkorišča podlaganje sporočila. Izvede se ga lahko pri nesimetrični šifri RSA, ki uporablja standard podlaganja PKCS#1, ki je prikazan na sliki 1.

Gradniki napada so: množica intervalov Mi, kriptopis c, izbrano število si, javni ključ, sestavljen iz vrednosti N in e, ter skrivni ključ, ki ga napadalec ne pozna, sestavljen iz d in N.

RSA definira zakriptirano sporočilo kot

$$c = m^e \mod N,$$

dekriptirano sporočilo pa kot:

$$m = c^d \mod N$$

Napad se lahko izvede, ker vemo, v katerem intervalu je m. Če označimo število

$$0x00 \quad 0x00 \quad 0x01 \quad \underbrace{0x00 \ldots 0x00}_{(k-2)}$$

z B kjer je k = dolžina sporočila + dolžina podloge, dobimo, da m obstaja med 2B in 3B − 1, saj je

$$2B = 0x00 \quad 0x00 \quad 0x02 \quad \underbrace{0x00 \ldots 0x00}_{(k-2)}$$

in

$$3B = 0x00 \quad 0x00 \quad 0x03 \quad \underbrace{0x00 \ldots 0x00}_{(k-2)}$$

Napadalec začne z izbiro števila s_0. S tem številom izračuna vrednost

$$s_i^e \cdot c \bmod N$$

in jo pošlje strežniku. Če strežnik odgovori, da sporočilo ustreza formatu PKCS#1, dobimo trditev:

$$2B \le s_i \cdot m \mod N \le 3B - 1$$

Če to pretvorimo v drugačno obliko, dobimo neenačbo:

$$2B \le s_i \cdot m - N \cdot r \le 3B - 1,$$

za vsak r, ki je celo število, kar smo dobili s pretvorbo formule za ostanek. Iz tega zdaj lahko izpeljemo naslednjo neenačbo:

$$\frac{2B + N \cdot r}{s_i} \le m \le \frac{3B - 1 + N \cdot r}{s_i},$$

torej smo dobili prvi interval za m. Iz prejšnje neenačbe lahko prav tako izpeljemo tole:

$$\frac{-3B + 1 + s_i \cdot m}{N} \le r \le \frac{-2B + s_i \cdot m}{N}$$

, pri čemer nastavimo vrednost m na levi strani na 2B, saj vemo, da ne more biti manjši od tega, in pa na desni na 3B − 1, saj vemo, da ne more biti večji od tega. Zdaj imamo vse potrebno, da izračunamo r. Iteriramo po vseh vrednostih, ki so znotraj intervala, kjer je r mogoč in vsako vstavimo v enačbo, kjer je izražen m. Vsakič, ko to naredimo dobimo nov interval za m. Nov interval se doda v interval množice intervalov.

Ko imamo prvi interval za m, se vrnemo na začetek, kjer napadalec pošilja izbrana števila. Ko je število pravilno, ponovimo celoten postopek, le da na koncu, preden dodamo nov interval v množico, preverimo, če ima neničelni presek s prejšnjo množico intervalov. Če je presek prazen, intervala ne dodamo, sicer pa ga dodamo. Na koncu upoštevamo novo množico intervalov in staro zavržemo.

Ko imamo v množici le še en interval oblike [a, a], vemo, da je a = m, torej smo našli dekriptirano sporočilo.

Ta napad si deli nekaj značilnosti s POODLE, na primer izkoriščanje podloge in ugibanje števil, v drugih pogledih pa se od njega precej razlikuje. [12][13]

0x00	0x02	Naključni bajti (podloga)	0x00	Čistopis

Slika 1: **Oblika sporočila s podlogo po standardu**

2.3 Asimetrična kriptografija

Da lahko začnemo kriptirati podatke z simetričnimi algoritmi kot je AES, najprej potrebujemo način dogovora o skrivnem ključu, ki ga bomo uporabljali pri šifriranju in dešifriranju. Eden od teh načinov je asimetrična kriptografija. Pri asimetrični kriptografiji imamo dva ključa: javni in skrivni ključ. Dva možna algoritma takšnega dogovora sta RSA in eliptične krivulje.

RSA je najširše uporabljena shema javne enkripcije ključa. Problem te metode je, da so potrebna zelo velika števila, da je varna. Priporočena dolžina ključa je 2048 bitov, kar je lahko za manjše, manj zmogljive naprave, problem. Da se temu problemu izognemo, lahko uporabimo eliptične krivulje.

Eliptična krivulja je definirana z enačbo

$$y^2 = x^3 + ax + b$$

in je simetrična glede na x os. Največja prednost, ki jo ima uporaba eliptičnih krivulj pred RSA je, da je osnovna operacija pri eliptičnih krivuljah seštevanje točk (angl. point addition), ki pa je zelo draga, kar pomeni, da imamo lahko veliko manjše ključe kot pri RSA. Če imamo enačbo $kP = E$ (kP kjer pomeni k-kratno seštevanje točke P samo s sabo) in poznamo vrednosti P in E, je iz tega zelo težko izračunati n. Najboljši algoritem za tak izračun, ki ga trenutno poznamo je Pollard's rho. Ta algoritem ima časovno kompleksnost $0(\sqrt{n})$, kar pomeni, da bi pri 256 bitnemu ključu k, potrebovali približno 10^{38} korakov, da bi ga ugotovili, kar pa je za današnje računalnike veliko preveč. Niso pa vse krivulje varne in jih moramo zato previdno izbrati. Na tak način lahko potem modificiramo na primer protokol Diffie-Hellman in izračunamo deljeno skrivnost. [14][15]

Protokol Diffie-Hellman z eliptičnimi krivuljami temelji na problemu diskretnega logaritma na eliptični krivulji, medtem ko navadni Diffie-Hellman temelji na problemu diskretnega logaritma. Ko dve strani (Bob in Alice) želita vzpostaviti varno komunikacijo prek javnega kanala, si morata izmenjati skupno skrivnost (angl. shared secret) tako, da je nihče ne more prestreči. Bob si zamisli število n, za katero velja $1 \leq n \leq red\ krivulje$. Izračuna $B = nP$, kjer je P javno znana točka na krivulji. Alice si zamisli število k, za katero veljajo enake omejitve kot n, in izračuna $A = kP$. Nato si izmenjata B in A. Zdaj lahko oba izračunata skupno skrivnost, Bob izračuna $A = kP$, Alice pa kB. Nekdo, ki bi izmenjavi prisluškoval, skrivnosti ne bi mogel izračunati, saj ne bi poznal vsaj ene skrivne vrednosti (n ali k). [16]

2.4 Pogoja za POODLE

Man-in-the-middle je napad, pri katerem napadalec pride med strežnik in odjemalca. Ko to uspešno naredi, lahko vidi šifrirana sporočila med njima, ne vidi pa čistopisov.

Za POODLE napad je to potrebno, saj napadalec potrebuje dostop do strežnika, ki pozna skrivni ključ in zna dešifrirati tajnopise, prav tako pa mora napadalec prestreči sporočila med odjemalcem in strežnikom. Napad se najpogosteje izvede z metodo "Spoofing". Poznamo več vrst, kot so: IP spoofing, ARP spoofing in DNS spoofing.

Pri IP spoofingu se napadalec predstavlja kot legitimna spletna stran, kar doseže s spreminjanjem IP paketov. Pri ARP spoofingu gre za povezovanje napadalčevega MAC naslova z IP naslovom legitimnega uporabnika. Ko odjemalec zdaj poskuša poslati sporočilo legitimnemu uporabniku, ga bo v resnici poslal napadalcu. Pri DNS spoofingu napadalec spremeni zapise na DNS strežniku in tako uporabnika preusmeri na svojo spletno stran, namesto na legitimno. [17]

V protoklih, ki so naprednejši od SSL 3.0 (npr. TLS 1.0), je potrebno strežnik prepričati, da za povezavo uporablja SSL 3.0, kjer bomo lahko izkoristili ranljivost POODLE.

To se doseže z izkoriščanjem **"downgrade dance"**. Downgrade dance je implementacija na nekaterih spletnih brskalnikih za lažjo kompatibilnost s starejšimi strežniki. Sproži se, če handshake ni uspel, saj odjemalec predvideva, da strežnik ne pozna protokola, s katerim odjemalec poskuša komunicirati. Nato odjemalec poskusi z nižjim protokolom in postopek ponovi. Napadalec torej v fazi handshake povzroči, da se ta ne izvede uspešno in to počne, dokler ne pride do protokola SSL 3.0, v katerem lahko izkoristi POODLE ranljivost [18].

3 SESTAVNI DELI NAPADA

3.1 AES-CBC

AES je simetrična šifra, ki se lahko uporablja v več načinih. Način, ki je pomemben za ta napad, je CBC (Cipher Block Chaining). CBC je način, v katerem AES deluje kot blokovna šifra. Ključna lastnost blokovne šifre je, da čistopis razreže na 16 bajtne bloke,

kar je potrebno pri šifriranju in dešifriranju. Pri šifriranju se nad vsakim blokom najprej izvede operacija xor (ekskluzivna disjunkcija) s prejšnjim šifriranim blokom, prvi blok pa za to operacijo uporabi IV (Initialization Vector). Vsak blok se torej pred šifriranjem pripravi takole: **encBlock[i-1] ^ ptBlock[i]**. [19] [20] Postopek dešifriranja je prikazan na sliki 2.

3.2 Podlaganje

Ker uporabljamo blokovno šifro, je potrebno zagotoviti, da je vsak blok dolg 16B, saj ne moremo izvesti xor operacije nad dvema blokoma različnih dolžin. To se doseže z dodajanjem podloge (angl. padding). V naši implementaciji uporabljamo standard PKCS#7, saj je najpogostejši standard podlaganja v AES-CBC.

Podloga po tem standardu se generira glede na to, koliko bajtov v bloku manjka do dolžine 16. Preostanek bajtov zapolni z bajti z vrednostjo števila mest, ki še manjkajo. Torej, če imamo blok, ki je dolg 13B, bodo zadnji trije bajti 0x03 0x03 0x03. Če je dolžina sporočila deljiva s 16 in ga lahko razrežemo na enako velike bloke, je treba dodati na konec še en blok dolžine 16B, ki bo imel vse bajte nastavljene na vrednost 0x10 (desetiško 16) [21].

3.3 Overitvena značka

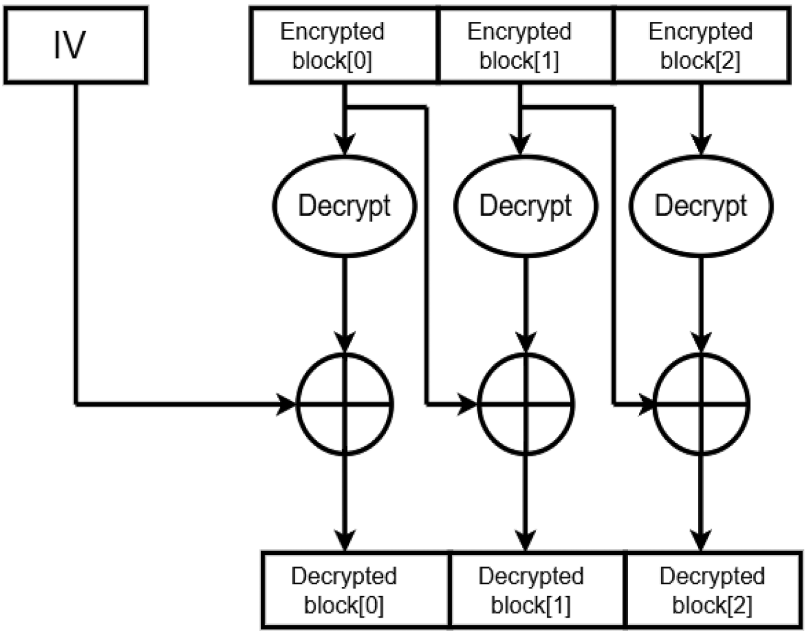
Za zagotavljanje celovitosti in avtentičnosti sporočila, se izračuna overitvena značka sporočila. To značko se nato priključi sporočilu, da jo lahko po dekripciji ponovno izračunamo in preverimo, če je enaka podani. Če je značka enaka, smo lahko prepričani, da se sporočilo ni spremenilo, v nasprotnem primeru pa avtentičnost sporočila ni več zagotovljena in ga je treba zavreči .

V naši implementaciji uporabljamo HMAC z zgoščevalno funkcijo SHA-256. Značko se izračuna pred kriptiranjem in pridruži čistopisu. Po koncu šifriranja se značko iz sporočila ponovno izračuna in primerja s tisto, ki je pritrjena na čistopis [21].

3.4 Overi-nato-šifriraj

Način overi-nato-šifriraj opisuje v kakšnem vrstnem redu se zgodi overitev sporočila. Najprej se izračuna značka iz čistopisa in se k njemu priključi. Nato sledi dodajanje podloge, ki se izračuna glede na dolžino čistopisa in značke skupaj. Vse skupaj se nato zašifrira.

Zaradi uporabe tega načina, se mora celotno sporočilo najprej dešifrirati, preveriti pravilnost podloge in šele nato preveriti veljavnost značke. To je še posebej problem pri blokovni šifri, saj imajo šifrirani bloki vpliv drug na drugega. To pomeni, da lahko spremenimo tajnopis in s tem dobimo informacije o čistopisu preden se značka preveri in strežnik sporočilo zaradi nepravilnosti zavrže. [21]



Slika 2: **Dešifriranje blokov pri AES-CBC**

Slika 3: **Struktura sporočila pri uporabi načina Overi-nato-šifriraj**

3.5 Operacija xor

Operacija xor je bitna operacija, ki vrne 0, če imata operanda oba enako vrednost, in ena, če imata različni. Operacija je komutativna in asociativna, kar je ključnega pomena pri POODLE napadu. Če torej izvedemo xor nad dvema bajtoma z isto vrednostjo, dobimo bajt 0x00, ki pa ne bo imel nobenega vpliva na katerokoli drugo vrednost v računu. Če pa operacijo izvedemo nad dvema različnima bajtoma, pa kot rezultat dobimo mesta kjer se razlikujeta (npr. **01100101 ^ 10010100 = 11110001**, saj se bajta razlikujeta na indeksih 0, 4, 5, 6 in 7, kjer je v rezultatu vrednost 1 in sta enaka na indeksih 1, 2, in 3, kjer je rezultat 0).

4 METODOLOGIJA

V programskem jeziku Java smo implementirali šifro AES-CBC, nad njo izvedli POODLE napad ter jo popravili tako, da napad ni več mogoč.

Pomembnejši knjižnici, ki smo ju uporabili pri implementaciji, sta crypto in security. Pri prvi smo uporabili Mac za overjanje sporočila in spec.SecretKeySpec, pri drugi pa SecureRandom za generiranje naključnega ključa in pa Inicializacijskega vektorja (IV).

4.1 Implementacija AES-CBC

Implementirali smo veriženje blokov po standardu CBC. Poleg tega smo implementirali tudi AES šifro s

- standardnimi operacijami:
- substitucija bajtov (angl. Byte substitution),
- zamik vrstic (angl. Shift rows),
- mešanje stolpcev (angl. Mix columns) in
- dodajanje rundnega ključa (angl. add Round key).

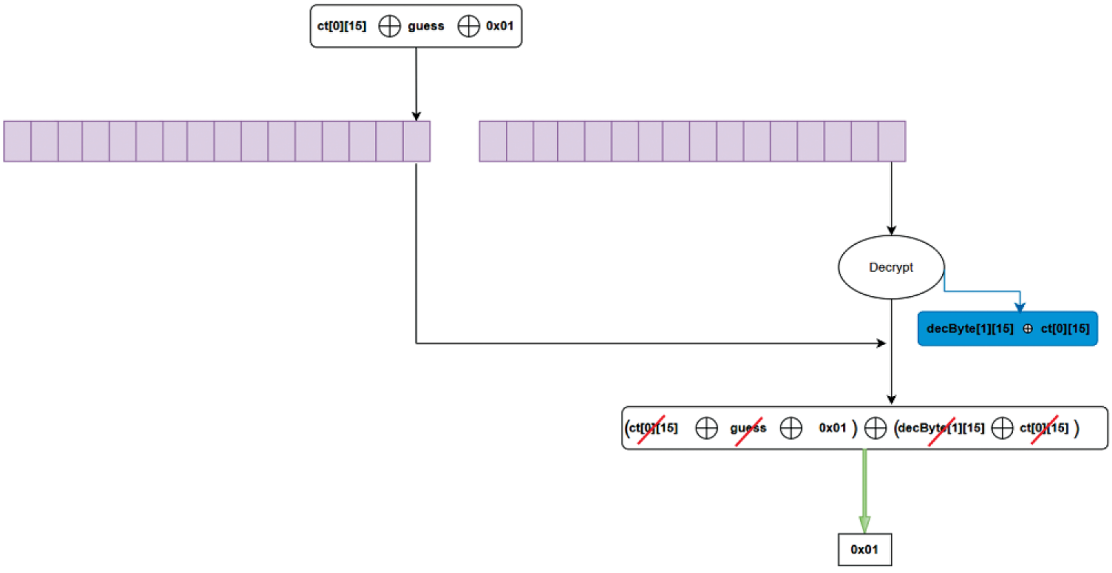
Matriki za substitucijo bajtov smo ročno zakodirali v program, enako smo naredili tudi pri matrikah za mešanje stolpcev in konstante rund.

Za overjanje sporočila smo uporabili HMAC z zgoščevalno funkcijo SHA-256 iz knjižnice crypto.

4.2 IPOODLE

Pri tem napadu ima napadalec na voljo kriptopis in strežnik, ki ga zna dešifrirati, a čistopisa ne pokaže. Čistopis je zakriptiran s šifro AES-CBC, narejen pa po načinu Overi-nato-šifriraj. Kot je opisano v poglavju AES-CBC, se nad vsakim blokom po šifriranju izvede xor s prejšnjim zakriptiranim blokom. Posledično se vsaka sprememba v določenem bloku odrazi tudi v naslednjem. Operacija XOR vrne vrednost 0, kadar sta oba bita enaka, pri tem pa velja, da se vsak bit, ki se xora z 0, ohrani (npr. **0x00 ^ 0x04 = 0x04**).

Napadalec začne s spreminjanjem zadnjega bajta predzadnjega bloka. Nad tem bajtom izvede xor z vrednostjo med 0 in 256 (0 vključeno) in z bajtom



Slika 4: **Dekriptiranje bloka pri POODLE napadu, če je zadnji bajt pravilno ugotovljen**

v vrednosti 0x01. Vrednost med 0 in 256 je vrednost, za katero napadalec ugiba, da je pravilna vrednost zadnjega bajta naslednjega bloka, 0x01 pa je vrednost podloge, ki se prilega poziciji mesta spremembe. Če je uganjena vrednost pravilna, bo strežnik vrnil napako podloge, saj se napadalčev bajt xor bajt naslednjega bloka ne bo izračunal v 0. Če pa je uganjena vrednost napačna, bo vrnil napako značke, saj je bila vrednost podloge pravilna (0x01 za zadnji bajt).

Ko je bajt pravilno ugotovljen, napadalec ponovi postopek na predzadnjem bajtu predzadnjega bloka, z razliko v vrednosti podloge, ki jo zdaj nastavi na 0x02. Popraviti je tudi potrebno prejšnji ugotovljeni bajt, da bo vrednost podloge tudi tam prišla 0x02. To lahko naredimo tako, da izvedemo xor med spremenjeno vrednostjo (**originalByte ^ guess ^ 0x01**) in (**changedByte ^ 0x01 ^ 0x02**), kjer se bosta bajta z vrednostjo 0x01 izničila, kar pomeni, da dobimo **originalByte ^ guess ^ 0x02**.

Ugotavljanje zadnjega bajta predzadnjega bloka je prikazano na sliki 4.

Ko napadalec ugotovi celoten blok, začne reševati blok pred tem. Vrednost podloge spet nastavi na 0x01, vrednosti prej spremenjenega bloka pa nastavi nazaj na originalne in shrani uganjene vrednosti v nov seznam. Ko spremeni vrednost bajta in je pripravljen na pošiljanje strežniku na dešifriranje, mora zadnji blok odstraniti, saj mora zdaj biti pravilna podloga na predzadnjem bloku in ne na zadnjem. Ker je lahko podložen le zadnji blok, mora biti blok, katerega vrednosti ugibamo, zadnji.

Težava nastane, ko je dešifriranih dovolj blokov, da ostanejo samo še trije. To je problem, saj so za pravilno dekripcijo potrebni vsaj štirje bloki. Prvi blok je vedno IV, naslednji bloki so besedilo, ki je bilo zakriptirano (število blokov je odvisno od dolžine besedila), naslednja dva bloka sta overitvena značka (v naši implementaciji uporabljamo HMAC s SHA-256, ki izračuna zgoščeno vrednost v dolžini 256 bitov, kar je 32 bajtov, torej 2 * 16) in na koncu še podloga, da dobimo dolžino deljivo s 16. Težavo se odpravi tako, da pred prvi blok dodamo (4 - numberOfRemainingBlocks) * 16B z vrednostjo 0x00. Ker imajo bajti vrednost 0, ne bodo imeli nobenega vpliva na vrednosti naslednjih blokov, bodo pa poskrbeli, da pri dekriptiranju ne bo prišlo do napake zaradi napačne dolžine.

Ko ugotovimo vrednosti vseh blokov razen prvega, je napad končan, saj je prvi blok IV, ki pa ni del sporočila, zato lahko ostane zakriptiran. Na koncu dobimo celoten dešifriran tajnopis. [21] [22]

5 ANALIZA

Implementirali smo dve rešitvi za preprečitev napada.
1. rešitev: Ena od možnosti preprečitve napada je, da imamo namesto dveh tipov napak, samo enega. Torej, če je podloga ali značka napačna, strežnik vedno vrne napako značke. Implementacija te rešitve je zelo pomembna, saj lahko napačna implementacija dopusti možnost časovnega napada. Pri časovnem napadu napadalec meri čas od trenutka, ko pošlje kriptopis strežniku, do trenutka, ko mu ta vrne napako. Če je čas med tema dvema dogodkoma manjši, lahko sklepa, da je bila podloga napačna, če pa je čas daljši, pa sklepa, da je bila podloga pravilna. Napačna implementacija te rešitve je lahko na primer takšna:

```
private static void checkPadding(byte[] pt) {
    int paddingLength = pt[pt.length - 1];
    if (paddingLength <= 0 || paddingLength > 16)
        throw new Report.InvalidTag();
    for (int i = pt.length - paddingLength; i < pt.length; i++) {
        if (pt[i] != paddingLength)
            throw new Report.InvalidTag();
    }
}
```

V prikazani implementaciji se napaka podloge sproži takoj, ko bajt ne ustreza podlogi, kar je veliko hitreje, kot če bi preverjali celotno podlogo in za tem še značko, zato je trivialno ugotoviti pravilnost podloge.
Pravilna implementacija izgleda takole:

```
private static void checkPadding(byte[] pt) {
    int paddingLength = pt[pt.length - 1];
    if (paddingLength <= 0 || paddingLength > 16)
        throwError = true;
    for (int i = pt.length - paddingLength; i < pt.length; i++) {
        if (pt[i] != paddingLength)
            throwError = true;
    }
}

private static void verifyTag(byte[] pt, byte[] tag, byte[] key) thr
ows Exception {
    final byte[] calculatedTag = createTag(pt, key);

    for (int i = 0; i < calculatedTag.length; i++) {
        if ((calculatedTag[i] ^ tag[i]) != 0)
            throwError = true;
    }
    if (throwError)
        throw new Report.InvalidTag();
}
```

static boolean throwError = false;
Pri preverjanju značke je potrebna posebna previdnost glede načina izvedbe, saj je preverjanje možno imple-

```
if (!Arrays.equals(tag, calculatedTag))
    throwError = true;

namesto:

for (int i = 0; i < calculatedTag.length; i++) {
    if ((calculatedTag[i] ^ tag[i]) != 0)
        throwError = true;
}
```

mentirati tudi na naslednji način:
Pri prvi rešitvi bi namreč lahko prišlo do časovnega napada.
2. rešitev: Še ena možna rešitev za obrambo pred POODLE napadom je, da namesto načina overi-nato-šifriraj uporabimo način šifriraj-nato-overi. Če uporabimo ta način, se vedno najprej preveri značka in šele nato tajnopis dešifriramo, kar bi preprečilo, da sploh pridemo do napake podloge, ker bi bilo že takoj ugotovljeno, da je bil tajnopis spremenjen, in bi ga zato strežnik zavrnil.

Na sliki 5 je prikazano, kateri deli kriptopisa so šifrirani pri uporabi načina Šifriraj-nato-overi. Dela, pobarvana z vijolično (čistopis in podloga), sta zakriptirana, medtem ko je del, pobarvan z zeleno

Slika 5: **Struktura sporočila pri uporabi načina Šifriraj-nato-overi**

(značka), v obliki čistopisa.

Časovna zahtevnost POODLE

Pri POODLE napadu za dešifriranje čistopisa potrebujemo največ 256 poskusov za dešifriranje enega bajta. Ker ugotavljamo vsak bajt posebej in dešifriranje enega bajta ne vpliva na dešifriranje drugega, se s povečevanjem dolžine sporočila potreben čas za izvedbo napada povečuje linearno, torej .

6 ZAKLJUČEK

V tem članku smo pokazali POODLE napad na šifro AES-CBC. Ugotovili smo, da sta kritični točki v SSL 3.0, ki naredita ta napad mogoč, način overi-nato-šifriraj in način CBC. S spremembo katere koli od teh dveh točk, napad ne bi bil več mogoč. Pokazali smo, da lahko dešifriramo celoten tajnopis, brez izjem, ne da bi poznali skrivni ključ.

Pred napadom smo se zavarovali na dva načina: strežnik je vedno vrnil napako značke in z uporabo načina šifriraj-nato-overi pri kriptiranju. Pri prvi rešitvi smo morali biti pazljivi na implementacijo, saj bi v primeru nepazljive izvedbe bil omogočen časovni napad, tako pri preverjanju pravilnosti značke kot tudi pri času vrnjene napake. Pri drugi rešitvi smo pokazali, da v načinu šifriraj-nato-overi napad ni mogoč, saj je najprej preverjena veljavnost značke, šele potem pa se začne postopek dekripcije.

Z napadom na šifro AES-CBC, ki uporablja način overi-nato-šifriraj smo uspeli dešifrirati celoten tajnopis. Pokazali smo, da ni potrebno veliko informacij za izvedbo napada. Vse kar potrebujemo je le sporočilo o tipu napake s strani strežnika (značke/podloge). Po popravkih načina šifriranja smo se napada tudi obranili, hkrati pa pazili, da implementacija ni odprla vrat za časovni napad.

Analiza je pokazala, da je ranljivost v SSL 3.0 kritična, saj napadalcu omogoča popolno dešifriranje podatkov, če pridobi dostop do strežnika za dekripcijo, kar je mogoče z izvedbo napada man-in-the-middle. Ker man-in-the-middle ni eden od enostavnejših napadov za izvesti, je tudi težko dobiti ustrezne pogoje za izvedbo POODLE, a ko enkrat te pogoje imamo, je napad trivialen.

Izpostavili smo, da je pri implementaciji zaščite potrebna posebna previdnost, saj lahko neustrezna izvedba povzroči nove ranljivosti. Pokazali smo, da je napad možen tudi pri uporabi protokolov, novejših od SSL 3.0, ali pri nepravilni implementaciji predlaganih rešitev.

Naše nadaljnje raziskave se osredotočajo na analizo različnih variacij POODLE napada na sodobnejše protokole, kot je TLS 1.3, ter na razvoj učinkovitih metod njihove preprečitve. Pravtako bi lahko raziskali podobne napade, zlasti tiste, ki izkoriščajo nepravilno ali nepopolno implementacijo zaščitnih ukrepov ter razviti mehanizme, ki bi večino tovrstnih napadov preprečile.

LITERATURA

[1] M. Ubaidullah and Q. Makki, “A Review on Symmetric Key Encryption Techniques in Cryptography,” International Journal of Computer Applications, vol. 147, no. 10, pp. 43–48, Aug. 2016, doi: 10.5120/ijca2016911203.

[2] D. Wagner and B. Schneier, “Analysis of the SSL 3.0 Protocol.”

[3] K. Bhargavan, C. Fournet, R. Corin, and E. Zalinescu, “Cryptographically verified implementations for TLS,” in Proceedings of the 15th ACM conference on Computer and communications security, Alexandria Virginia USA: ACM, Oct. 2008, pp. 459–468. doi: 10.1145/1455770.1455828.

[4] J. Zhou, W. Fu, W. Hu, Z. Sun, T. He, and Z. Zhang, “Challenges and Advances in Analyzing TLS 1.3-Encrypted Traffic: A Comprehensive Survey,” Electronics, vol. 13, no. 20, p. 4000, Oct. 2024, doi: 10.3390/electronics13204000.

[5] E. A. Fischer, “Cybersecurity Issues and Challenges.”

[6] J. M. Biju, N. Gopal, and A. J. Prakash, “CYBER ATTACKS AND ITS DIFFERENT TYPES,” vol. 6, no. 3, 2019.

[7] Y. Li and Q. Liu, “A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments,” Energy Reports, vol. 7, pp. 8176–8186, Nov. 2021, doi: 10.1016/j.egyr.2021.08.126.

[8] S. Gohwong, “The State of the Art of Cryptography-Based Cyber-Attacks,” Asian Crime and Society Review, vol. 6, no. 2, pp. 24–34, Jul. 2019, Accessed: Sep. 10, 2025. [Online]. Available: https://so02.tci-thaijo.org/index.php/IJCLSI/article/view/242595

[9] W. Duo, M. Zhou, and A. Abusorrah, “A Survey of Cyber Attacks on Cyber Physical Systems: Recent Advances and Challenges,” IEEE/CAA Journal of Automatica Sinica, vol. 9, no. 5, pp. 784–800, May 2022, doi: 10.1109/JAS.2022.105548.

[10] Y. Gluck, N. Harris, and A. Ngel, “BREACH: REVIVING THE CRIME ATTACK.”

[11] P. G. Sarkar and S. Fitzgerald, “ATTACKS ON SSL A COMPREHENSIVE STUDY OF BEAST, CRIME, TIME, BREACH, LUCKY 13 & RC4 BIASES.”

[12] D. Bleichenbacher, “Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1,” in Advances in Cryptology — CRYPTO ’98, H. Krawczyk, Ed., Berlin, Heidelberg: Springer, 1998, pp. 1–12. doi: 10.1007/BFb0055716.

[13] H. Bock, J. Somorovsky, and C. Young, “Return Of Bleichenbacher’s Oracle Threat (ROBOT).”

[14] E. Bach, “Toward a theory of Pollard’s rho method,” Information and Computation, vol. 90, no. 2, pp. 139–155, Feb. 1991, doi: 10.1016/0890-5401(91)90001-I.

[15] V. Kapoor, V. S. Abraham, and R. Singh, “Elliptic curve cryptography,” Ubiquity, vol. 2008, no. May, pp. 1–8, May 2008, doi: 10.1145/1386853.1378356.

[16] R. Haakegaard and J. Lang, “The Elliptic Curve Diffie-Hellman (ECDH).”

[17] A. Mallik, "MAN-IN-THE-MIDDLE-ATTACK: UNDERSTANDING IN SIMPLE WORDS."
[18] E. S. Alashwali and K. Rasmussen, "What's in a Downgrade? A Taxonomy of Downgrade Attacks in the TLS Protocol and Application Protocols Using TLS," in Security and Privacy in Communication Networks, vol. 255, R. Beyah, B. Chang, Y. Li, and S. Zhu, Eds., Cham: Springer International Publishing, 2018, pp. 468–487. doi: 10.1007/978-3-030-01704-0_27.
[19] National Institute of Standards and Technology (US), "Advanced Encryption Standard (AES)," National Institute of Standards; Technology (U.S.), Washington, D.C., NIST FIPS 197-upd1, May 2023. doi: 10.6028/NIST.FIPS.197-upd1.
[20] J. Daemen, "The Rijndael Block Cipher."
[21] D. Jelenc, "Varnost podatkov." Accessed: Apr. 21, 2025. [Online]. Available: https://varnost-podatkov.lem.im/6-ae-kdfs.html?print-pdf#/sec-title-slide
[22] B. Möller, T. Duong, and K. Kotowicz, "This POODLE Bites: Exploiting The SSL 3.0 Fallback."

▪

Anja Klančar je študentka na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Zanimajo jo področja kriptografije, kibernetske varnosti in umetne inteligence.

▪

Matevž Pesek je izredni profesor in raziskovalec na Fakulteti za računalništvo in informatiko Univerze v Ljubljani, kjer je diplomiral (2012) in doktoriral (2018). Od leta 2009 je član Laboratorija za računalniško grafiko in multimedije. Od leta 2024 izvaja predmeta Varnost programov in Varnost sistemov, kjer se raziskovalno ukvarja s poučevanjem konceptov in organizacijo dogodkov s področja računalniške varnosti.

Analiza spletnih strani slovenskih univerz z vidika optimizacije za iskalnike

Marko Urh, Eva Jereb, Alenka Baggia
Fakulteta za organizacijske vede, Univerza v Mariboru, Kidričeva cesta 55a, 4000 Kranj
marko.urh@um.si, eva.jereb@um.si, alenka.baggia@um.si

Izvleček

Univerzitetne spletne strani so pomembne pri zagotavljanju spletne prisotnosti univerz. Obiskujejo jih bodoči in trenutni študenti, visokošolski učitelji in sodelavci, raziskovalci, partnerji in drugi ter predstavljajo blagovno znamko univerze, referenčno točko za spletne strani fakultet ter so vir informacij za obiskovalce. Kljub temu pa številne univerze še vedno nimajo vzpostavljenih sistematičnih načinov optimizacije za iskalnike, kar se odraža v pomanjkljivo zasnovanih spletnih straneh z vidika optimizacije za iskalnike. Namen raziskave je bil analizirati spletne strani slovenskih univerz z vidika optimizacije za iskalnike, opozoriti na pomanjkljivosti ter podati predloge za izboljšave. Ugotavljamo, da imajo obravnavane spletne strani številne pomanjkljivosti. Največ izzivov predstavljajo sklop, vezan na hitrost nalaganja spletne strani, in tehnične pomanjkljivosti na sami spletni strani. Zelo dobro pa so spletne strani prilagojene za mobilne naprave. Podana so priporočila za spremljanje in urejanje spletnih strani z vidika optimizacije za iskalnike. Ugotovitve prispevajo k boljšemu razumevanju pomena optimizacije spletnih strani ter posledično boljšo prisotnost in vidnost na spletu.

Ključne besede: analiza spletnih strani, optimizacija spletnih strani, optimizacija za iskalnike, slovenske univerze, spletna vidnost

Analysis of Slovenian university websites from the perspective of search engine optimization

Abstract

University websites are essential in ensuring the online presence of universities. They are visited by future and current students, higher education teachers and associates, researchers, partners, and others and represent the university's brand, a reference point for faculty websites and a source of information for visitors. Nevertheless, many universities still do not have established systematic methods for search engine optimization, which is reflected in inadequately designed websites from an SEO perspective. The purpose of the research was to analyze the websites of Slovenian universities from the perspective of search engine optimization, point out shortcomings, and provide suggestions for improvements. We conclude that all websites under consideration, have numerous shortcomings. The biggest challenges are related to the speed of website loading and technical shortcomings on the website itself. However, the websites are very well adapted for mobile devices. Recommendations are given for monitoring and editing websites from the perspective of web optimization for search engines. The findings contribute to a better understanding of the importance of website optimization and, consequently, to improved online presence and visibility.

Keywords: web page analysis, web page optimization, search engine optimization, Slovenian universities, online visibility

1 UVOD

Spletne strani univerz obiskujejo različni uporabniki, kot so bodoči študenti, ki iščejo informacije o študiju, programih, pogojih za vpis [1], trenutno vpisani študenti, ki dostopajo do informacij o predmetih ter administrativnih storitev [2], akademsko in administrativno osebje, družinski člani študentov ali bodočih študentov, ki iščejo informacije o programih, univerzitetni politiki [3]the research aims to discern the extent to which universities facilitate two-way communication with stakeholders through digital platforms. Methodology: Websites were selected based on world university rankings, encompassing institutions from each continent. Websites were selected based on global university rankings, spanning institutions across continents. Qualitative content analysis employed predefined and emergent categories to evaluate interactivity and organizational listening features on these platforms. Findings reveal consistent stakeholder mapping but significant disparities in communication tools and channels, which impacts real-time, asynchronous, and symmetric engagement effectiveness. Communication structures range from integrated governance roles to fragmented responsibilities, influencing stakeholder accessibility and institutional transparency. Discussion: implications for organizational communication practices are discussed, highlighting strategies to enhance stakeholder engagement via institutional websites. The study underscores the pivotal role of communication management teams in fostering transparency and responsiveness. Conclusions: advocating for leveraging technological advancements, conclusions propose transforming websites into proactive platforms for organizational listening. Recommendations emphasize developing tailored communication strategies to optimize engagement and effectiveness in Higher Education contexts.\n , \n Introducción: Este estudio investiga métodos de escucha organizacional en la enseñanza superior mediante análisis comparativo de sitios web de principales universidades globales y las tres mejores de Lituania. Se enfatiza el rol de estos sitios como plataformas digitales y evalúa cómo facilitan la comunicación bidireccional con partes interesadas. Metodología: Se seleccionaron los sitios web según clasificaciones mundialmente conocidas, incluyendo instituciones de cada continente. El análisis cualitativo utilizó categorías predefinidas y emergentes para evaluar interactividad y características de escucha organizativa. Resultados: Se identificó un mapeo consistente de partes interesadas, con disparidades en herramientas y canales que afectan la participación efectiva en tiempo real, asíncrona y simétrica. Estructuras de comunicación variaron de roles integrados a responsabilidades fragmentadas, influenciando accesibilidad y transparencia. Discusión: Se abordan implicaciones para prácticas de comunicación organizacional, enfocándose en soluciones para mejorar el compromiso a través de sitios web institucionales. Se destaca el papel crucial de equipos de gestión de comunicación. Conclusiones: Al promover el aprovechamiento de avances tecnológicos, se propone transformar los sitios web en plataformas proactivas para la escucha organizacional, recomendando el desarrollo de estrategias de comunicación adaptadas para optimizar el compromiso y la eficacia en contextos de educación superior.", "container-title": "European Public & Social Innovation Review", "DOI": "10.31637/epsir-2024-576", "ISSN": "2529-9824", "journalAbbreviation": "epsir", "license": "http://creativecommons.org/licenses/by-nc-nd/4.0", "page": "1-21", "source": "DOI.org (Crossref, lokalni prebivalci, raziskovalci, podjetja, udeleženci dogodkov in drugi [1].

Uporabniki iščejo in dostopajo do informacij na svetovnem spletu na različne načine. Kljub porastu iskanj s pomočjo orodij generativne umetne inteligence (ChatGPT, DeepSeek, Gemini, Perplexity in drugi) se velika večina (več kot 90 %) uporabnikov še vedno poslužuje spletnih iskalnikov (angl. Search Engine) [4], [5]. Podatki o uporabi spletnih iskalnikov kažejo, da med vsemi iskalniki bistveno izstopa Google, ki ga za iskanje informacij uporablja kar 89,57 % uporabnikov, sledijo Bing s 4,02 %, Yandex 2,19 %, Yahoo! 1,49 %, DuckDuckGo 0,95 % in drugi [6]. Uporabniki najpogosteje izberejo povezave, ki so na prvi strani iskalnih rezultatov. Na prvi strani se med rezultati iskanja najpogosteje nahajajo tako imenovani organski rezultati iskanja (angl. Organic Search Results) in plačljivi rezultati, med uporabniki pa so najbolj priljubljeni in zaupanja vredni organski rezultati iskanja. Zato je ključnega pomena, da se spletna stran uvršča na prvo stran zadetkov oz. čim višje. Višje, kot se spletna stran pojavi v rezultatih iskanja, več obiskovalcev bo kliknilo in obiskalo izbrano spletno stran [7]. Za organske rezultate iskanja pa je značilno, da so to neplačani zadetki (niso rezultat oglaševanja) in predstavljajo večino spletnega pro-

meta [8]. Število organskih zadetkov ter posledično vidnost in obiskanost spletnih strani univerz lahko bistveno izboljšamo z ustrezno optimizacijo spletne strani. Pri tem ima ključno vlogo optimizacija za iskalnike ali SEO (angl. Search Engine Optimization).

V digitalni dobi je zato za univerze nujno, da razumejo in poskrbijo za svojo spletno prisotnost in vidnost, saj je le-ta pomembna za informiranje in vpis bodočih študentov [9], informiranje in komunikacijo s pedagoškimi delavci, raziskovalci, administracijo in drugimi [10], skrb za podobo univerze in njeno blagovno znamko [11], konkurenčnost na trgu ter dobro uvrščenost na različnih lestvicah [12]. Na pomen spletne prisotnosti univerz opozarja tudi Nacionalna agencija Republike Slovenije za kakovost v visokem šolstvu (NAKVIS). Na njihovi spletni strani lahko najdemo dve globalni metriki, ki poudarjata pomen spletne prisotnosti univerz [13]:

- Webometrics Ranking ocenjuje spletno prisotnost in vpliv univerz. Webometrics Ranking za merjenje spletne prisotnosti uporablja metodologijo, ki spletne strani rangira po štirih kriterijih: (1) vidnost – število zunanjih povezav (angl. backlinks) z drugih spletnih mest na domeno univerze; (2) transparentnost – število citiranih raziskovalcev; (3) odličnost – število najbolj citiranih znanstvenih objav; (4) spletna vidnost ter raziskovalna odličnost.
- UniRank poudarja pomen spletne prisotnosti in razvršča univerze po spletni pomembnosti in vidnosti. Ključni kriteriji metodologije UniRank temeljijo na neodvisnih spletnih meritvah, ki so: (1) obiskanost spletne strani univerze; (2) prisotnost in dejavnost na družbenih omrežjih; (3) število zunanjih povezav; (4) avtoriteta domene; (5) iskalna vidnost in optimizacija za iskalnike (SEO).

Cilj raziskave in članka je analizirati spletne strani slovenskih univerz z vidika SEO, primerjati spletne strani, ugotoviti, katere spletne strani univerz potrebujejo največ nadgradenj z vidika SEO, opozoriti in predstaviti SEO kazalnike, ki so najbolj problematični ter kateri sklopi kazalnikov potrebujejo največ pozornosti.

Članek v nadaljevanju predstavlja teoretična izhodišča s poudarkom na SEO. Sledi predstavitev univerz in pomena njihovih spletnih strani z vidika SEO, ki vpliva na vidnost in prisotnost na spletu. Sledita predstavitev metod dela s poudarkom na procesu zbiranja podatkov, ki smo ga izvedli s spletnim orodjem, ter predstavitev obdelave in analize podatkov. Nato so predstavljeni rezultati raziskave, diskusija in sklep.

2 PREGLED LITERATURE

Spletna stran (angl. web page) je dokument, zapisan v jeziku HTML (angl. Hyper Text Markup Language), ki je dostopen na spletu. Spletišče ali spletno mesto (angl. web site) pa predstavlja skupek med seboj pomensko povezanih spletnih strani [14].

Začetna oz. vstopna spletna stran predstavlja najpomembnejšo spletno stran za obiskovalce nekega spletnega mesta in deluje kot glavna vstopna točka, preko katere uporabniki začnejo svoj obisk na spletnem mestu [15]. Začetna oziroma vstopna spletna stran je ključni element spletnega mesta zaradi svoje edinstvene vloge pri navigaciji in interakciji z uporabniki, njen glavni namen pa je pritegniti pozornost obiskovalcev, vzpostaviti odnos ter spodbuditi nadaljnjo interakcijo [16]during a single web navigation, capture customers' attention, build rapport, and prompt them to act. By showing how to capture customer commitment over the course of a single website visit, the concept of customer website engagement, defined as "the process of developing cognitive, affective and behavioural commitment to an active relationship with the website", addresses strategic concerns. Drawing from literature on engagement, the purpose of this paper is to consider how retail websites can engage customers during the course of a website navigation. A conceptual model of website customer engagement underpinned by relationship marketing and communication knowledge, shows how perceptions of the website's exploration and sense-making potential can activate consumer engagement, and is then empirically tested.\n \n \n Design/methodology/approach\n \n Using survey data, measures of the four dimensions of engagement (interaction engagement, activity engagement, behavioural engagement, and communication engagement.

Začetna ali vstopna spletna stran beleži največ obiska v primerjavi z drugimi stranmi na spletnem mestu, kar je posledica dejstva, da začetna stran služi kot glavna vstopna točka za uporabnike, ki na spletno mesto prihajajo preko rezultatov iskanj v spletnih iskalnikih, družbenih medijev ali neposrednih povezav [17], [18]. Običajno je zasnovana tako, da predstavlja najbolj dostopen in privlačen del spletnega mesta, kjer ponuja ključne informacije ali povezave, ki obiskovalce usmerjajo do drugih delov spletnega

mesta [18]. Njen namen so usmerjanje in navigacija obiskovalcev, predstavitev organizacije, spodbujanje aktivnosti in krepitev občutka zaupanja pri obiskovalcih, nudenje osnovnih informacij in drugo. Zaradi vsega naštetega velja, da je začetna ali vstopna spletna stran ena najpomembnejših spletnih strani in je dober pokazatelj splošne kakovosti nekega spletnega mesta.

Formanek [19] opredeljuje optimizacijo za iskalnike (SEO) kot sistematičen pristop, s katerim lahko izboljšamo obseg in kakovost spletnega prometa, kar se doseže z optimizacijo vsebine in strukture spletnega mesta z namenom doseganja višjih pozicij v rezultatih iskanj. Optimizacija za iskalnike se najpogosteje izvaja na dveh ravneh, in sicer na spletni strani (angl. On-Page SEO) in izven spletne strani (angl. Off-Page SEO). Optimizacija na spletni strani vključuje tehnike optimizacije spletne strani neposredno na spletni strani [20]. Na kazalnike na spletni strani lahko vplivamo sami. O optimizaciji izven spletne strani pa govorimo takrat, ko skušamo vplivati na kazalnike, ki so zunaj same spletne strani. Takšna optimizacija se osredotoča predvsem na dvigovanje prepoznavnosti v obliki gradnje povezav, dejavnosti na družbenih omrežjih in drugo [21].

S sistemskim izvajanjem spletne optimizacije lahko – podobno kot ostale organizacije – univerze vplivajo na rezultate spletnih iskalnikov (npr. organski zadetki, slike, videoposnetki, lokalno iskanje), kar vpliva na število obiskovalcev in prepoznavnost blagovne znamke [22]. Univerze, ki se zavedajo pomena SEO in aktivno izvajajo tehnike SEO na svojih spletnih straneh, poročajo o dvigu spletnih obiskov ter boljši uvrstitvi na spletnih iskalnikih [12].

Raziskave kažejo, da obstaja močna povezava med uspešnostjo SEO univerzitetnih spletnih strani in njihovo spletno prepoznavnostjo [23]a method was devised that quantified the website quality and search engine optimization (SEO. Kljub temu pa raziskovalci ugotavljajo, da imajo spletne strani univerz še veliko priložnosti za izboljšanje [24], [25]. Analiza kazalnikov SEO, kot so število povratnih povezav, kakovost vsebine in tehnična optimizacija, razkriva, da te metrike neposredno vplivajo na uvrstitev univerz v spletnih iskalnikih. Kot ugotavljajo [12], se pri optimizaciji spletnih strani lahko učinkovito uporabijo tudi veliki jezikovni modeli (angl. Large Language Models, LLM).

Kot ugotavljajo v raziskavi [23]a method was devised that quantified the website quality and search engine optimization (SEO, rangiranje na seznamu 100 univerz ARWU (angl. Academic Ranking of World Universities) pokaže, da obstaja zmerna povezava med akademsko odličnostjo in kakovostjo začetne spletne strani, medtem ko performansi SEO niso povezani s kakovostjo. Podobno kot v svetu je tudi v Sloveniji, kjer deluje več univerz, SEO učinkovitost spletnih strani univerz še dokaj neraziskana. V prispevku [26] sicer najdemo nekaj priporočil za izboljšavo optimizacije spletnih strani fakultet, kljub temu pa večina ustanov optimizaciji ne posveča posebne pozornosti. Na osnovi tega smo izvedli raziskavo, osredotočeno na učinkovitost SEO spletnih strani slovenskih univerz.

3 METODOLOGIJA

Za zbiranje podatkov smo uporabili spletno orodje SEO Site Checkup. SEO Site Checkup velja za uveljavljeno in priznano spletno orodje, ki je bilo uporabljeno v številnih raziskavah, povezanih z optimizacijo in izboljšavami spletnih strani [27] ter izvajanjem primerjalnih študij [23]a method was devised that quantified the website quality and search engine optimization (SEO. Omenjeno orodje velja za koristno zaradi odsotnosti omejitev uporabe in velikega števila meritev kazalnikov, ki vplivajo na SEO [12]. Orodje je znano po hitrem delovanju in natančnih poročilih, ki so osnova za začetek odprave težav, kar lahko omogoči izboljšanje SEO optimizacije [19], zato smo se v raziskavi odločili za uporabo orodja SEO Site Checkup.

S pomočjo orodja SEO Site Checkup smo analizirali 74 kazalnikov SEO (v prilogi so našteti vsi kazalniki v angleškem jeziku, s slovenskimi prevodi, razporejeni po sklopih in po univerzah, skupaj z izmerjenimi vrednostmi), ki se pojavljajo na začetni spletni strani posamezne univerze. Kazalniki so razdeljeni v pet sklopov:

- V sklop »Pogoste SEO težave (angl. Common SEO Issues)« so uvrščeni osnovni tehnični in drugi kazalniki (26 kazalnikov), ki so gradniki spletne strani.
- Sklop »Optimizacija hitrosti (angl. Speed Optimizations)« je povezan s kazalniki (25 kazalnikov), ki imajo vpliv na hitrost nalaganja spletne strani.
- Sklop »Strežnik in varnost (angl. Server and Security)« predstavlja kazalnike (10 kazalnikov), ki jih najpogosteje najdemo na strežniku in njegovih nastavitvah, ter druge kazalnike, povezane z varnostjo.

- Sklop »Prilagodljivost za mobilne naprave (angl. Mobile Usability)«. Prilagojenost za mobilne naprave je pomembna, saj se določen del spletnega prometa odvija preko mobilnih naprav (tablični računalniki, pametni telefoni, prenosni računalniki …) (trije kazalniki).
- V sklop »Napredni SEO (angl. Advanced SEO)« sodijo kazalniki (10 kazalnikov), ki so povezani z naprednejšimi tehničnimi vidiki in zahtevami spletne strani.

Vsak kazalnik se v analizi kvalitativno ovrednoti z vrednostmi: PASSED, WARNING, FAILED ali STATUS ONLY. Pomen vrednosti lahko interpretiramo kot: PASSED (vrednost kazalnika na spletni strani je prisotna oziroma ustrezno nastavljena), WARNING (vrednost kazalnika na spletni strani ni optimalno nastavljena in bi zahtevala manjši popravek), FAILED (vrednost kazalnika na spletni strani manjka oziroma je neustrezno nastavljena) in STATUS ONLY (vrednost kazalnika je prikazana zgolj informativno). Primeri možnih stanj kazalnikov:

- PASSED – primer: datoteka Robots.txt obstaja ali pa ne obstaja. Če obstaja, potem se kazalnik ovrednoti kot PASSED.
- WARRNINGS – primer: pri kazalniku Image Alt imajo slike na spletni strani opredeljene lastnosti Image Alt, vendar so brez vsebine oz. besedila.
- FAILED – primer: datoteka Robots.txt obstaja ali pa ne obstaja. Če ne obstaja, potem se kazalnik ovrednoti kot FAILED.
- STATUS ONLY – primer: »Mobile Snapshot« prikaže videz spletne strani, kot bi le-ta izgledala na mobilni napravi.

Za potrebe raziskave je bilo treba zbrati podatke, jih analizirati, oceniti, primerjati in dobronamerno opozoriti na pomanjkljivosti, ki se pojavljajo na spletnih straneh slovenskih univerz z vidika SEO. Dobro optimizirana spletna stran za iskalnike vpliva na spletno vidnost, pomembnost in prisotnost univerze na spletu. Na pomen naštetega opozarjajo številni avtorji in organizacije [13], [28], [29]. V okviru raziskave smo skušali odgovoriti na štiri raziskovalna vprašanja:

- RV1: Katere spletne strani slovenskih univerz imajo največ priložnosti za izboljšave z vidika SEO?
- RV2: Kateri SEO kazalniki spletnih strani slovenskih univerz predstavljajo največje izzive?
- RV3: Kateri sklopi SEO kazalnikov predstavljajo največje izzive za slovenske univerze?
- RV4: Spletna stran katere slovenske univerze je najmanj prilagojena z vidika hitrosti nalaganja?

V prvem delu raziskave smo izvedli kvantitativno analizo, v okviru katere smo ugotavljali stanje kazalnikov SEO, v drugem delu pa smo izvedli primerjavo med spletnimi stranmi.

Seznam slovenskih univerz in spletnih naslovov (tabela 1) smo pridobili z uradne spletne strani NAKVIS, ki je v Sloveniji zadolžena za akreditiranje visokošolskih zavodov in študijskih programov.

Po preverjanju pravilnosti spletnih naslovov univerz smo pridobili podatke o začetni oz. vstopni spletni stran vsake univerze, in sicer s pomočjo spletnega orodja SEO Site Checkup (brezplačna verzija) [30]. Kot je bilo že omenjeno, velja začetna ali vstopna spletna stran za eno najpomembnejših in reprezentativnih spletnih strani glede splošne kakovosti nekega spletišča. Za zagotavljanje enakih pogojev smo vse podatke zbrali v enem dnevu (28. 5. 2025). Podatke iz poročil, ki jih generira orodje, smo prepisali v

Tabela 1: **Slovenske univerze s kraticami in spletnimi naslovi [13].**

Naziv univerze (po abecednem vrstnem redu)	**Spletni naslov (URL)**
Evro-sredozemska univerza (EMUNI)	https://emuni.si/
Nova univerza (NU)	https://www.nova-uni.si/
Univerza Alma Mater Europaea (AMEU)	https://www.almamater.si/
Univerza na Primorskem (UP)	https://www.upr.si/
Univerza v Ljubljani (UL)	https://www.uni-lj.si/
Univerza v Mariboru (UM)	https://www.um.si/
Univerza v Novem mestu (UNM)	https://uni-nm.si/
Univerza v Novi Gorici (UNG)	https://ung.si/

preglednico. V raziskavi smo dobljene kvalitativne vrednosti (»PASSED«, »WARNING«, »FAILED« in »STATUS ONLY«) pretvorili v številčne, in sicer 1, 2, 3 in 4 (PASSED=1; WARRNINGS=2; FAILED=3 in STATUS ONLY=4). Tako smo standardizirali podatke in omogočili kvantitativno obdelavo. Pravilnost vnesenih podatkov v tabeli je bila zagotovljena z večkratnim preverjanjem vnesenih podatkov s strani avtorjev prispevka.

4 REZULTATI

Prvo raziskovalno vprašanje se je glasilo: »Katere spletne strani slovenskih univerz imajo največ priložnosti za izboljšave z vidika SEO?« Rezultate primerjalne analize in odgovor na raziskovalno vprašanje lahko razberemo iz tabele 2. Na vsaki spletni strani je bilo analiziranih 74 kazalnikov, ki vplivajo na optimizacijo spletne strani. Največje število kazalnikov, katerih vrednosti na spletni strani manjkajo oziroma so neustrezno nastavljene, ima spletna stran Univerze Alma Mater Europaea, in sicer 25, štirje kazalniki na tej spletni strani pa bi potrebovali manjši popravek. Sledita ji spletna stran Univerze v Novi Gorici z 18 kazalniki, katerih vrednosti na spletni strani manjkajo oz. so neustrezno nastavljene, in s štirimi kazalniki, ki bi potrebovali manjši popravek. Spletna stran Univerze v Ljubljani ima 15 kazalnikov, katerih vrednosti na spletni strani manjkajo oz. so neustrezno nastavljene, in sedem kazalnikov, ki bi potrebovali manjši popravek. Najmanj kazalnikov, katerih vrednosti na spletni strani manjkajo oz. so neustrezno nastavljene (11), najdemo na spletni strani Univerze v Novem mestu. Zaključimo lahko, da ima največ priložnosti za izboljšavo spletne strani z vidika SEO spletna stran Univerze Alma Mater Europaea.

Drugo raziskovalno vprašanje je bilo vezano na kazalnike spletnih strani univerz, ki predstavljajo največje izzive z vidika SEO. Ker je bilo analiziranih kazalnikov 74, smo se odločili, da zaradi lažje preglednosti in razumevanja predstavimo najpogostejših pet kazalnikov (po sklopih) (tabela 3), katerih vrednosti na spletni strani manjkajo oz. so neustrezno nastavljene.

Kot je razvidno iz tabele 3, se v sklopu Pogoste SEO težave (angl. Common SEO Issues) pojavljata dva velika izziva, ki se nanašata na neustrezno prilagojenost slik na posamezni spletni strani za različne naprave (računalnik, tablica, pametni telefon), in sicer ločljivost zaslona (angl. Responsive Image) in neustrezna uporaba CSS stilov v HTML dokumentu (angl. Inline CSS), ki se pojavita pri sedmih spletnih straneh. Na petih straneh najdemo neustrezno rabo ključnih besed (angl. Keywords Usage) in pomanjkljivosti glede strukturiranosti, berljivosti in optimizacije spletnih naslovov (angl. SEO Friendly URL). Pri štirih spletnih straneh pa se pojavijo neustrezne nastavitve in uporaba zemljevida spletnega mesta, ki se najpogosteje nahaja v XML datoteki (angl. Sitemap).

Analiza kazalnikov v sklopu Optimizacije hitrosti (angl. Speed Optimizations) pokaže, da sta pri vseh osmih spletnih straneh problematična dva kazalnika zaradi svojih vrednosti. Prvi kazalnik je povezan s testiranjem pridobivanja spletnih elementov spletne strani, ki so najpogosteje slike, CSS datoteke, Ja-

Tabela 2: **Število kazalnikov, katerih vrednosti na spletni strani manjkajo ali so neustrezno nastavljene (FAILED), ter njihov odstotek in število kazalnikov, katerih vrednosti niso optimalno nastavljene ter zahtevajo manjši popravek (WARNINGS) skupaj z njihovim odstotkom.**

Univerza	Število kazalnikov, katerih vrednosti na spletni strani manjkajo ali so neustrezno nastavljene (FAILED).	Število kazalnikov, katerih vrednosti niso optimalno nastavljene in zahtevajo manjši popravek (WARNINGS).
Univerza Alma Mater Europaea	25 (34 %)	4 (5 %)
Univerza v Novi Gorici	18 (24 %)	4 (5 %)
Univerza v Ljubljani	15 (20 %)	7 (9 %)
Evro-sredozemska univerza	13 (18 %)	6 (8 %)
Univerza v Mariboru	13 (18 %)	4 (5 %)
Nova univerza	13 (18 %)	4 (5 %)
Univerza na Primorskem	12 (16 %)	6 (8 %)
Univerza v Novem mestu	11 (15 %)	3 (4 %)

Tabela 3: **Število kazalnikov, katerih vrednosti na spletni strani manjkajo ali so neustrezno nastavljene – razdeljeno po sklopih.**

Sklop	Kazalnik	Število kazalnikov, katerih vrednosti na spletni strani manjkajo ali so neustrezno nastavljene.
Pogoste težave SEO	Responsive image	7
	Inline CSS	7
	Keywords Usage	5
	SEO Friendly URL	5
	Sitemape	4
Optimizacije hitrosti	Page Objects	8
	Render Blocking Resources	8
	Modern Image Format	6
	Site Loading Speed	5
	Image Metadata	3
	JavaScript Caching	3
	Largest Contentful Paint	3
Strežniki in varnostjo	Plaintext Emails	6
	HSTS	5
	Unsafe Cross-origin Links	5
	Server Signature	3
	HTTP/2	2
Prilagodljivost za mobilne naprave	Meta Viewport	1
Napredni SEO	Structured Data	5
	Ads.txt Validation	1
	Custom 404 Error Page	1

vaScript datoteke in drugo (angl. Page Objects). Drugi kazalnik je povezan z datotekami, ki jih mora brskalnik prenesti in obdelati pred prikazom strani na zaslonu (angl. Render Blocking Resources). S šestimi kazalniki na spletni strani sledi težava, povezana z uporabo starejših formatov slik na spletni strani (angl. Modern Image Format). Pet spletnih strani ima težave glede hitrosti nalaganja spletne strani (angl. Site Loading Speed). Tri spletne strani imajo težave z metapodatki slik (angl. Image Metadata), tri strani imajo težave s shranjevanjem JavaScript datotek na strani brskalnika (angl. JavaScript Caching) ter tri strani s časom, potrebnim za prikaz največjega elementa spletne strani na zaslon (npr.: slike) (angl. Largest Contentful Paint).

Kazalniki, uvrščeni v sklop, ki je povezan s Strežniki in varnostjo (angl. Server and Security), predstavljajo pomemben del pojavnosti spletne strani. Največ težav (šest strani) imajo spletne strani univerz z nezaščitenimi e-poštnimi naslovi, ki so zapisani kot običajno besedilo (angl. Plaintext Emails). Težava tovrstnega zapisa e-naslova je v tem, da ga lahko prebere tudi poštni pajek (angl. Spam bot). Pri petih straneh se pojavi problem pri varnostnem kazalniku, ki preverja, ali spletna stran uporablja mehanizem za dostopanje preko varne povezave (angl. HSTS – HTTP Strict Transport Security). Prav tako na petih straneh zasledimo varnostna tveganja glede povezav, ki kažejo na zunanje spletne naslove brez ustrezne zaščite (angl. Unsafe Cross-Origin Links). Problematične vrednosti kazalnika Server Signature se pojavljajo pri treh univerzah, HTTP/2 pa pri dveh univerzah. Kazalnik Server Signature preverja posredovanje informacij o strežniku (verzija, tip …) (angl.

Server Signature Test). HTTP/2 pa je kazalnik, ki preverja uporabo protokola HTTP/2 na spletni strani. Protokol HTTP/2 se uporablja za prenos podatkov na relaciji brskalnik–strežnik.

Sklop Prilagodljivost za mobilne naprave (angl. Mobile Usability) ima najmanj napak z vidika spletne optimizacije. Ugotavljamo, da obstaja samo en kazalnik pri eni univerzitetni spletni strani, ki je problematičen. To je kazalnik, ki preverja, ali je spletna stran prilagojena za prikazovanje na mobilnih napravah Meta Viewport.

Pozitivno so presenetila tudi stanja kazalnikov v sklopu Naprednega SEO (angl. Advanced SEO). V tem sklopu najbolj problematično izstopajo vrednosti kazalnika Structured Data, ki se pojavi pri petih spletnih straneh. Omenjen kazalnik se nanaša na pravilno vključevanje strukturiranih podatkov na spletni strani, ki spletnim iskalnikom omogočajo ustreznejši pregled vsebine in prikaz rezultatov. Kot problematično se stanje enkrat pojavi pri kazalniku Ads.txt, drugič pa pri Custom 404 Error Page. Prvi kazalnik je povezan s preverjanjem ustrezne nastavitve datoteke ads.txt (avtorizacija za prodajalce oglasov) (angl. Ads.txt Validation). Kazalnik Custom 404 Error Page se nanaša na manjkajočo spletno stran, ki uporabniku na prijazen način sporoča, da iskana spletna stran ne obstaja.

Odgovor na tretje raziskovalno vprašanje, ki se glasi »Kateri sklopi SEO kazalnikov predstavljajo največje izzive za slovenske univerze?«, razkrije številne razlike med univerzami. V tabeli 4 predstavljamo število kazalnikov na posamezni univerzitetni spletni strani, po posameznih sklopih, katerih vrednosti manjkajo oziroma so neustrezno nastavljene (angl. FAILED). Tako lahko vidimo (po vrsticah), da so takšni kazalniki na spletni strani Evro-sredozemske univerze (EMUNI) trije, in sicer v sklopu Pogoste SEO težave. V sklopu Optimizacija hitrosti jih je šest, v sklopu Strežnik in varnost štirje, medtem ko v sklopih Prilagodljivost za mobilne naprave in Napredni SEO ni nobenega, skupaj torej 13. Seštevek kazalnikov po sklopih (po stolpcih) pokaže, da je v sklopu Pogoste SEO težave seštevek vseh kazalnikov, katerih vrednosti na spletni strani manjkajo oziroma so neustrezno nastavljene (angl. FAILED), vseh osmih univerz 40. V sklopu Optimizacija hitrosti 50, v sklopu Strežnik in varnost 22, v sklopu Prilagodljivost za mobilne naprave eden in v sklopu Napredni SEO sedem. Izračun odstotka kazalnikov, katerih vrednosti na spletni strani manjkajo oziroma so neustrezno nastavljene po posameznem sklopu glede na vse kazalnike v posameznem sklopu pri vseh univerzah, je izračunan z enačbo (1). Izračunani odstotki so prikazani v tabeli 4 v zadnji vrstici. Odstotki kazalnikov, katerih vrednosti na spletni strani manjkajo oziroma so neustrezno nastavljene, po posameznem sklopu so sledeči: 19 % v sklopu Pogoste težave, 25 % v sklopu Optimizacija hitrosti, 28 % v sklopu Strežnik in varnost, 3 % v sklopu Prilagodljivost za mobilne naprave ter 9 % v sklopu Napredni SEO.

Tabela 4: **Število kazalnikov, katerih vrednosti na spletni strani manjkajo oziroma so neustrezno nastavljene (angl. FAILED) – po sklopih in odstotki le-teh.**

	SKLOP					
Univerza (kratica)	**Pogoste SEO težave (D1=26)**	**Optimizacija hitrosti (D2=25)**	**Strežnik in varnost (D3=10)**	**Prilagodljivost za mobilne naprave (D4=4)**	**Napredni SEO (D5=10)**	**Skupaj**
EMUNI	3	6	4	0	0	13
NU	4	6	2	0	1	13
AMEU	7	11	3	1	3	25
UP	4	5	3	0	1	12
UL	6	5	3	0	1	15
UM	3	8	2	0	0	23
UNM	4	3	3	0	1	11
UNG	9	6	3	0	0	18
Skupaj (FX) ‚FAILED'	F1=40 (19 %)	F2=50 (25 %)	F3=22 (28 %)	F4=1 (3 %)	F5=7 (9 %)	

Odstotek "FAILED" kazalnikov v sklopu =

$$\frac{F_x}{U \times D_x} \times 100\ \%$$

F_X = skupno število vseh 'FAILED' kazalnikov spletnih strani univerz v enem sklopu
U = število univerz = 8
D_X = število kazalnikov v sklopu

V sklopu raziskave smo skušali odgovoriti še na četrto raziskovalno vprašanje, ki se glasi: »Spletna stran katere univerze je najmanj prilagojena z vidika hitrosti nalaganja spletne strani?« Za potrebe analize hitrosti nalaganja spletne strani univerz je bilo uporabljenih 25 kazalnikov. V tabeli 5 je prikazano število kazalnikov, katerih vrednosti na spletni strani manjkajo oziroma so neustrezno nastavljene (angl. FAILED), in število kazalnikov, katerih vrednosti na spletni strani niso optimalno nastavljene in bi zahtevale manjši popravek (angl. WARNINGS).

Največ kazalnikov, katerih vrednosti na spletni strani manjkajo oziroma so neustrezno nastavljene, je povezanih s prilagojenostjo z vidika hitrosti nalaganja spletne strani in jih najdemo na strani Univerze Alma Mater Europaea, in sicer 11. En kazalnik na tej strani pa bi potreboval manjši popravke. Najmanj kazalnikov, katerih vrednosti na spletni strani manjkajo oziroma so neustrezno nastavljene, pa najdemo na spletni strani Univerze v Novem mestu, in sicer tri.

5 DISKUSIJA

Rezultati raziskave kažejo, da imajo vse spletne strani univerz večje ali manjše pomanjkljivosti pri optimizaciji za spletne iskalnike. Pomanjkljivosti smo ugotovili na samih spletnih straneh (angl. On-Page SEO) in izven spletnih strani (angl. Off-Page SEO). Ugotavljamo, da je z vidika SEO najbolj prilagojena spletna stran Univerze v Novem mestu, najmanj pa spletna stran Univerze Alma Mater Europaea. Optimizacija za iskalnike SEO velja za eno izmed strategij, s katero lahko izboljšamo položaj uvrstitve spletne strani v iskalnikih. Na ta način lahko izboljšamo število organskega spletnega prometa [31]websites have increasingly relied on digital marketing practices, notably search engine optimization (SEO, zato je nujno potrebno, da je število kazalnikov, katerih vrednosti na spletni strani manjkajo oziroma so neustrezno nastavljene, čim manjše oziroma jih ni, kar lahko vpliva na vidnost v iskalnikih in zadovoljstvo uporabnikov.

Največ kazalnikov, katerih vrednosti na spletni strani manjkajo oziroma so neustrezno nastavljene z vidika SEO, ima spletna stran Univerze Alma Mater Europaea. Od 74 analiziranih kazalnikov je takšnih 25, kar predstavlja približno 34 % vseh kazalnikov. Za omenjeno spletno stran pa ugotavljamo, da ima tudi štiri kazalnike, ki bi zahtevali manjši popravek, kar predstavlja približno 5 %. Druga spletna stran s številnimi izzivi je stran Univerze v Novi Gorici z 18 kazalniki, katerih vrednosti na spletni strani

Tabela 5: **Število kazalnikov, katerih vrednosti na spletni strani manjkajo ali so neustrezno nastavljene (FAILED), ter njihov odstotek, in število kazalnikov, katerih vrednosti niso optimalno nastavljene ter zahtevajo manjši popravek (WARNINGS), skupaj z njihovim odstotkom z vidika hitrosti nalaganja spletne strani.**

Univerza (kratica)	Število kazalnikov, katerih vrednosti na spletni strani manjkajo ali so neustrezno nastavljene (FAILED).	Število kazalnikov, katerih vrednosti niso optimalno nastavljene in zahtevajo manjši popravek (WARNINGS).
AMUE	11 (42 %)	1 (4 %)
UM	8 (31 %)	2 (8 %)
UMENI	6 (23 %)	2 (8 %)
UNG	6 (23 %)	2 (8 %)
NU	6 (23 %)	1 (4 %)
UL	5 (19 %)	3 (12 %)
UP	5 (19 %)	3 (12 %)
UNM	3 (12 %)	1 (4 %)

manjkajo oziroma so neustrezno nastavljene (približno 24 %), in s štirimi kazalniki, ki bi zahtevali manjši popravek. Sledi spletna stran Univerze v Ljubljani, ki ima 15 kazalnikov, katerih vrednosti na spletni strani manjkajo oziroma so neustrezno nastavljene (približno 20 %), ter sedem kazalnikov, ki bi zahtevali manjši popravek.

Najbolje uvrščena spletna stran oz. stran z najmanj kazalniki, katerih vrednosti na spletni strani manjkajo oziroma so neustrezno nastavljene, je spletna stran Univerze v Novem mestu, ki ima 11 takšnih kazalnikov (približno 15 %). Druga najbolje uvrščena je spletna stran Univerze na Primorskem z 12 kazalniki (približno 16 %), sledi ji spletna stran Nove univerze s 13 kazalniki, katerih vrednosti na spletni strani manjkajo oziroma so neustrezno nastavljene (približno 18 %). Ugotavljamo, da ima spletna stran Univerze v Novem mestu več kot 2-krat manj kazalnikov, katerih vrednosti na spletni strani manjkajo oziroma so neustrezno nastavljene, kot spletna stran Univerze Alma Mater Europaea, kar je pohvalno, še vedno pa ostaja potreba po izboljšanju. Povprečno število kazalnikov, katerih vrednosti na spletni strani manjkajo oziroma so neustrezno nastavljene, na spletnih straneh slovenskih univerz pa je 15 oz. približno 20 %. Ugotavljamo, da je približno vsak peti kazalnik, ki ima vpliv na spletno optimizacijo, potreben korekcije. Če upoštevamo še opozorila na spletnih straneh, pa je potreba po korekcijah še večja.

Ugotavljamo, da se na vseh osmih spletnih straneh pojavljata dva kazalnika, in sicer Page Objects in Render Blocking Resources, katerih vrednosti so problematične. SEO Site Checkup [32] omenja, da se s kazalnikom Page Objects preverja možnost pridobivanja vseh podatkov na spletni strani. Če pridobivanje ni mogoče, se lahko zgodi, da se stran ne prikaže pravilno. To pa lahko povzroči slabšo uporabniško izkušnjo in nižjo uvrstitev v iskalnikih. Kazalnik Render Blocking Resources povezujemo s preverjanjem nekaterih datotek (npr.: JavaScript, CSS …) oz. njihovega blokiranja pred prikazom na zaslonu. Odprava nepravilnosti, povezane s tem kazalnikom, lahko bistveno izboljša hitrost nalaganja spletne strani. Omenjeni neustreznosti nakazujeta na določeno stopnjo pomanjkanja izkušenj pri samem načrtovanju in izdelavi spletnih strani, njihovem vzdrževanju in pomanjkanju sistematičnega spremljanja spletnih strani z vidika spletne optimizacije.

Nadaljnje ugotavljamo, da obstajata dva kazalnika pri sedmih spletnih straneh univerz, katerih vrednosti na spletni strani manjkajo oziroma so neustrezno nastavljene. To sta Responsive Image in Inline CSS. S kazalnikom Responsive Image se preveri, ali so slike na spletni strani ustrezne velikosti za prikazovanje na uporabnikovem zaslonu. Uporabniki dostopajo do spletnih strani z različnih naprav (prenosni računalniki, pametni telefoni, tablični računalniki …), kar veča potrebo in pomen po prilagajanju slik. Vsako neustrezno prikazovanje in prilagajanje pa podaljša čas nalaganja strani, kar pa posledično slabša uporabniško izkušnjo. Kazalnik Inline CSS priporoča, da se lastnosti spletne strani ne vgrajujejo v HTML dokument, temveč se opredelijo izven HTML dokumenta. S tem lahko izboljšamo čas nalaganja spletne strani, kar pa spet pripomore k boljši uporabniški izkušnji.

Zanimive rezultate smo dobili tudi z analizo sklopov kazalnikov, ki vplivajo na optimizacijo za iskalnike. V ta namen smo analizirali 74 kazalnikov, ki vplivajo na SEO in so v petih sklopih. Sledijo našteti sklopi s številom kazalnikov v njih: pogoste SEO težave (26 kazalnikov), optimizacije hitrosti (25 kazalnikov), strežniki in varnost (10 kazalnikov), prilagodljivost za mobilne naprave (4 kazalniki) in napredni SEO (10 kazalnikov).

Ker se število kazalnikov med posameznimi sklopi razlikuje, nas je zanimal delež (v odstotkih) kazalnikov, katerih vrednosti na spletni strani manjkajo ali so nepravilno nastavljene v posameznem sklopu. Rezultati analize in odgovor na zastavljeno vprašanje pokaže, da imajo spletne strani univerz največ težav na področju strežniških nastavitev in varnosti. Ugotavljamo, da je v tem sklopu kar 28 % kazalnikov, katerih vrednosti na spletni strani manjkajo ali so nepravilno nastavljene. Drugi sklop, kjer smo zaznali največ težav, je povezan z optimizacijo hitrosti nalaganja spletne strani, kjer je kazalnikov, katerih vrednosti na spletni strani manjkajo ali so nepravilno nastavljene, kar 25 %. Težave, povezane z optimizacijo za iskalnike, zasledimo tudi v sklopu pogostih težav za iskalnike (19 %), ki so najpogosteje povezani z nepravilnimi nastavitvami določenih elementov (npr.: Responsive Image, Inline CSS, Keywords Usage, SEO Friendly URL in drugimi). Relativno malo kazalnikov, katerih vrednosti na spletni strani manjkajo ali so nepravilno nastavljene, zasledimo na področju naprednega SEO, kjer je takšnih kazalnikov približno 7 %. Najbolj pozitivne rezultate

pa zaznavamo na področju prilagodljivosti za mobilne naprave, kjer je kazalnikov, katerih vrednosti na spletni strani manjkajo ali so nepravilno nastavljene, približno 3 %. Na področju razvoja spletnih strani je že dolgo znano, da se spletne strani, prilagojene za mobilne naprave, bolje uvrščajo na rezultatih iskanj (npr.: v Googlu), kar velja od aprila 2015 [33].

V okviru raziskave nas je zanimalo, kako so spletne strani univerz prilagojene z vidika hitrosti, ki je eden izmed pomembnejših kazalnikov. Med drugim pozitivno vpliva na uporabniško izkušnjo [34], zaupanje v blagovno znamko [35], optimizacijo za iskalnike [36], učinkovitost in stroške [37], dostopnost [23]a method was devised that quantified the website quality and search engine optimization (SEO, manjšo porabo energije ter posledično na ogljični odtis. Analiziranih je bilo 25 kazalnikov, vsak izmed njih pa ima določen vpliv na hitrost nalaganja spletne strani. Povprečno število kazalnikov, katerih vrednosti na spletni strani manjkajo ali so nepravilno nastavljene, z vidika hitrosti nalaganja spletne strani, je 6,25 (25 %). Najmanj prilagojeno spletno stran z vidika hitrost ima Univerza Alma Mater Europaea, kjer je kar 11 kazalnikov, katerih vrednosti na spletni strani manjkajo ali so nepravilno nastavljene (približno 42 %). Najbolje se na področju hitrosti izkaže spletna stran Univerze v Novem mestu. Omenjena stran ima samo tri kazalnike, katerih vrednosti na spletni strani manjkajo ali so nepravilno nastavljene (približno 12 %). Ugotavljamo, da ima najbolje uvrščena univerza kar 3,6-krat manj kazalnikov, katerih vrednosti na spletni strani manjkajo ali so nepravilno nastavljene z vidika hitrosti, kot najslabše uvrščena univerza.

Analiza pokaže in opozori na številne nepravilnosti, ki jih imajo spletne strani slovenskih univerz z vidika SEO. Številne spletne strani univerz so zgrajene na sistemih za upravljanje vsebin (angl. Content Management System), med katerimi so najbolj poznani WordPress, Joomla!, Drupal, Shopify in Wix. Veliko število neustreznih kazalnikov SEO vzbuja sum, da se v omenjenih sistemih ne uporabljajo vtičniki (angl. Plugins), ki podpirajo SEO. Razlike med univerzami so lahko posledica uporabe različnih sistemov za upravljanje vsebin, pomanjkanje politike upravljanja spletnih strani na ravni univerze, finančnih ter kadrovskih omejitev ali drugih dejavnikov. Pogosto pa je vidik SEO sistemsko zapostavljen oz. temelji na znanju in samoiniciativnosti osebe ali oseb, ki so na univerzi zadolžene za upravljanje spletnih strani.

Raziskava ima določene omejitve. Dejstvo je, da se spletne strani vsebinsko neprestano spreminjajo. Dobljeni rezultati tako predstavljajo stanje spletnih strani v določenem trenutku. Zato bi bilo smiselno in priporočljivo izvajati obdobna testiranja in analize spletnih strani z vidika SEO. Omenjen predlog predstavlja tudi idejo za nadaljnje delo oz. prihodnje raziskave. Z ustreznim razumevanjem spletne optimizacije in njenih tehnik lahko dokaj hitro izboljšamo spletno stran (angl. On-Page SEO). Optimizacija zunaj spletne strani (angl. Off-Page SEO) pa lahko zahteva več napora in časa.

Pri analizi smo opazili, da ni bilo mogoče testirati štirih kazalnikov na spletni strani Univerze v Mariboru, kar predstavlja približno 0,7 % vseh vrednosti analiziranih kazalnikov. Najverjetneje je to posledica določene zaščite proti računalniškim programom (programski robot ali bot), požarnega zidu (angl. Firewall) ali druge zaščite. Višja stopnja zaščite je verjetno posledica kibernetskega napada, ki se je zgodil oktobra 2024 na omenjeni univerzi.

Zavedamo se tudi, da je analiza temeljila na uporabi enega spletnega orodja (SEO Site Checkup), ki je sicer mednarodno priznano in uporabljeno tako v poslovnem kot akademskem področju. Večina sorodnih orodij (Seobility, SEOptimer, WebsiteSeoCheckersicer in drugi) temelji na podobnih tehnikah preverjanja kazalnikov z vidika SEO. Kot primer navajamo kazalnik Robots.txt, ki ga analizira večina spletnih orodij. Ta kazalnik obstaja ali pa ne obstaja in je z vidika analize dokaj enostavno določljiv, kot tudi številni drugi kazalniki (npr.: Meta Title, Meta Description, Heading Tags, Sitemap, SSL Checker, HTTPS, HTTP2, Responsive Image, Noindex Tag).

Na osnovi pridobljenih spoznanj priporočamo, da univerze začnejo sistematično spremljati spletne strani z vidika SEO. To pomeni sprejetje določenih aktivnosti, kot so redno izvajanje SEO testov, spremljanje dobrih praks in drugo. Učinkovita politika, povezana s spletno optimizacijo, mora upoštevati strateške načrte in cilje univerze, njene posebnosti in povratne informacije uporabnikov [25]. Nikakor ne smemo zanemariti rednih izobraževanj s področja SEO, kar priporočajo tudi drugi avtorji [25]. Če obstajajo finančne možnosti, bi bilo smiselno najeti določene strokovnjake ali podjetja, ki se ukvarjajo z izboljšanjem SEO kazalnikov.

V prihodnosti bi bilo smiselno izvesti raziskavo, kjer bi primerjali spletne strani slovenskih univerz

z nekaterimi evropskimi z vidika SEO. Prav tako bi bila priporočljiva analiza spletnih strani univerz, ki imajo večino kazalnikov ustreznih oz. se dobro uvrščajo na lestvicah SEO. Takšna analiza lahko služi kot primer dobre prakse. Prav tako bi bilo smiselno raziskavo razširiti oz. povezati tudi z drugimi kazalniki, kot so čas, preživet na posamezni spleti strani, število obiskanih strani ter podatki o spletni dostopnosti (angl. Web Accessibility).

6 SKLEP

Čas, v katerem živimo, zahteva od večine organizacij prisotnost in vidnost na spletu, ki pa ju lahko izboljšamo z določenimi pristopi spletne optimizacije za iskalnike. Večina iskanj se namreč začne prav preko iskalnikov (npr.: Google). Izvedena raziskava se osredotoča na analizo spletnih strani slovenskih univerz z vidika prilagojenosti za iskalnike (SEO). Kljub temu da gre za spletne strani podobnih organizacij, obstajajo med njimi precejšnje razlike z vidika SEO.

Ugotavljamo, da prav vse spletne strani univerz potrebujejo izboljšave oz. nobena ni popolnoma ustrezna z vidika SEO, kar je zaskrbljujoče. Z vidika števila kazalnikov, katerih vrednosti na spletni strani manjkajo ali so nepravilno nastavljene, najbolj izstopa spletna stran Univerze Alma Mater Europaea, ki ima največ težav v sklopu Pogoste SEO težave in Optimizacije hitrosti.

Kar zadeva prilagojenost spletnih strani glede časa nalaganja oz. optimizacije hitrosti, ugotavljamo, da ima najmanj kazalnikov, katerih vrednosti na spletni strani manjkajo ali so nepravilno nastavljene, Univerza v Novem mestu. Največ kazalnikov, katerih vrednosti na spletni strani manjkajo ali so nepravilno nastavljene, pa najdemo pri Univerzi Alma Mater Europaea. Na osnovi raziskave ugotavljamo, da je najbolje prilagojena spletna stran Univerza v Novem mestu, najslabše pa spletna stran Univerze Alma Mater Europaea.

Na osnovi rezultatov raziskave ugotavljamo pomanjkanje sistematičnega pristopa univerz z vidika spletne optimizacije. Dejstvo je, da si slovenske univerze medsebojno konkurirajo in zato ostaja potreba po čim večji prisotnosti in vidnosti na spletu. Hkrati pa se morajo zavedati dejstva, da v svetu predstavljajo Slovenijo kot državo. S sistematičnim pristopom k SEO bi lahko vse slovenske univerze izboljšale svoj položaj in s tem dvignile stopnjo prisotnosti, vidnosti in ugled sebi in Sloveniji. Zato priporočamo, da se na ravni univerz uvede sistematičen pristop k SEO. Univerze pa lahko pridobljeno znanje in izkušnje prenašajo na svoje članice oz. fakultete.

LITERATURA

[1] T. Heino, S. Rauti, R. Carlsson, in V. Leppänen, "Third-party services as a privacy threat on university websites", v *Proceedings of the 24th International Conference on Computer Systems and Technologies*, Ruse Bulgaria: ACM, jun. 2023, str. 134–138. doi: 10.1145/3606305.3606335.

[2] A. Paterson, "What is a Library Website, Anyway? Reconsidering Dominant Conceptual Models", *Partnership*, let. 16, št. 1, str. 1–22, jul. 2021, doi: 10.21083/partnership.v16i1.6363.

[3] M. Sueldo, "(How) do universities listen? Evidence from institutional websites of the world's top universities", *epsir*, let. 9, str. 1–21, sep. 2024, doi: 10.31637/epsir-2024-576.

[4] S. Sarkar, "Are AI Chatbots Replacing Search Engines?", onelittleweb. Pridobljeno: 7. november 2025. [Na spletu]. Dostopno na: https://onelittleweb.com/data-studies/ai-chatbots-vs-search-engines

[5] M. Kapuściński, "LLM-Powered Search vs Traditional Search: 2025-2030 Forecast", TMS. Pridobljeno: 7. november 2025. [Na spletu]. Dostopno na: https://ttms.com/llm-powered-search-vs-traditional-search-2025-2030-forecast/#1.--google-still-crushes-ai-tools-in-search-volume

[6] StatCounter, "StatCounter GlobalStats", Search Engine Market Share Worldwide. Pridobljeno: 16. junij 2025. [Na spletu]. Dostopno na: https://gs.statcounter.com/search-engine-market-share

[7] N. Surana, D. M. Gala, in R. U. Kanthe, "Impact on Website Traffic due to Google Algorithm Update", *EPRA*, let. 9, št. 1, str. 258–262, jan. 2023, doi: 10.36713/epra12259.

[8] K. I. Roumeliotis, N. D. Tselikas, in C. Tryfonopoulos, "Greek Hotels' Web Traffic: A Comparative Study Based on Search Engine Optimization Techniques and Technologies", *Digital*, let. 2, št. 3, str. 379–400, jul. 2022, doi: 10.3390/digital2030021.

[9] A. Al-Qahtani, "Website Representations of Saudi Universities in Makkah Region: A Critical Discourse Analysis Approach", *Cogent Arts & Humanities*, let. 8, št. 1, str. 1895463, jan. 2021, doi: 10.1080/23311983.2021.1895463.

[10] E. Orduña-Malea in J.-A. Ontalba-Ruipérez, "Proposal for a multilevel university cybermetric analysis model", *Scientometrics*, let. 95, št. 3, str. 863–884, jun. 2013, doi: 10.1007/s11192-012-0868-5.

[11] I. Budnikevych, O. Kolomytseva, in D. Bastrakov, "Communication component in the formation of the image of higher education institutions based on a marketing approach", *ebcstu*, let. 24, št. 4, str. 5–16, nov. 2023, doi: 10.62660/ebcstu/4.2023.05.

[12] K. Król, M. Jaworska, in D. Goshchynska, "SEO auditing using large language models as a key university marketing strategy component", *SPSUTOM*, let. 2024, št. 207, 2024, doi: 10.29119/1641-3466.2024.207.18.

[13] NAKVIS, Podatki o visokošolskih zaodih. Pridobljeno: 15. maj 2025. [Na spletu]. Dostopno na: https://nakvis.si/analize-in--publikacije/porocila-strokovnjakov-in-odlocbe

[14] iSlovar, "http://www.islovar.org/islovar", iSlovar. Pridobljeno: 29. oktober 2025. [Na spletu]. Dostopno na: http://www.islovar.org/islovar

[15] V. Trulock in R. Hetherington, "Assessing the Progress of Implementing Web Accestibility - An Irish Case Study":, v *Proceedings of the Tenth International Conference on Enterprise Information Systems*, Barcelona, Spain: SciTePress - Science and and Technology Publications, 2008, str. 105–111. doi: 10.5220/0001667001050111.

[16] C. Demangeot in A. J. Broderick, "Engaging customers during a website visit: a model of website customer engagement", *IJRDM*, let. 44, št. 8, str. 814–839, avg. 2016, doi: 10.1108/IJRDM-08-2015-0124.

[17] Y. Helmy, A. E. Khedr, S. Koleif, in E. Mohammed Haggag, "Adaptive Approach for Intelligent Web to Enhance Business Intelligence Applications", *The International Journal of Informatics Bulletin*, let. 1, št. 1, str. 20–28, jan. 2019, doi: 10.21608/fcihib.2019.107509.

[18] W. Aqeel, B. Chandrasekaran, A. Feldmann, in B. M. Maggs, "On Landing and Internal Web Pages: The Strange Case of Jekyll and Hyde in Web Performance Measurement", v *Proceedings of the ACM Internet Measurement Conference*, Virtual Event USA: ACM, okt. 2020, str. 680–695. doi: 10.1145/3419394.3423626.

[19] M. Formanek, "Solving SEO Issues in DSpace-based Digital Repositories: A Case Study and Assessment of Worldwide Repositories", *ITAL*, let. 40, št. 1, mar. 2021, doi: 10.6017/ital.v40i1.12529.

[20] K. Kowalczyk in T. Szandala, "Enhancing SEO in Single-Page Web Applications in Contrast With Multi-Page Applications", *IEEE Access*, let. 12, str. 11597–11614, 2024, doi: 10.1109/ACCESS.2024.3355740.

[21] R. Badr in H. Maher, "Evaluating governmental tourism websites in Egypt using Search Engine Optimization tools", *International Journal of Heritage, Tourism and Hospitality*, let. 13, št. 2, str. 286–298, sep. 2019, doi: 10.21608/ijhth.2019.132241.

[22] F. S. Labausa, J. M. Pinca, in N. E. Cruda, "Investigating Digital Marketing Strategies in Influencing Student Enrollment Decisions in Tertiary Education", *CJBIS*, let. 5, št. 5, str. 119–133, sep. 2023, doi: 10.34104/cjbis.023.01190133.

[23] A. Giannakoulopoulos, N. Konstantinou, D. Koutsompolis, M. Pergantis, in I. Varlamis, "Academic Excellence, Website Quality, SEO Performance: Is there a Correlation?", *Future Internet*, let. 11, št. 11, str. 242, nov. 2019, doi: 10.3390/fi11110242.

[24] A. S. Halibas, A. M. Cherian, I. G. Pillai, L. B. Reazol, E. G. Delvo, in G. H. Sumondong, "Web Ranking of Higher Education Institutions: An SEO Analysis", v *2020 International Conference on Computation, Automation and Knowledge Management (ICCAKM)*, 2020, str. 411–415. doi: 10.1109/ICCAKM46823.2020.9051481.

[25] M. Vállez in A. Ventura, "Analysis of the SEO visibility of university libraries and how they impact the web visibility of their universities", *The Journal of Academic Librarianship*, let. 46, št. 4, str. 102171, jul. 2020, doi: 10.1016/j.acalib.2020.102171.

[26] M. Urh in A. Baggia, "Spletne strani fakultet Univerze v Mariboru z vidika spletne optimizacije", v *39th International Conference on Organizational Science Development*, University of Maribor Press, 2020, str. 871–887. doi: 10.18690/978-961-286-388-3.68.

[27] A. S. Sarlis, I. C. Drivas, in D. P. Sakas, "Implementation and Dynamic Simulation Modeling of Search Engine Optimization Processes. Improvement of Website Ranking", v *Strategic Innovative Marketing*, A. Kavoura, D. P. Sakas, in P. Tomaras, Ur., v Springer Proceedings in Business and Economics. , Cham: Springer International Publishing, 2017, str. 437–443. doi: 10.1007/978-3-319-56288-9_57.

[28] B. Dolai, S. J. Shenmare, in V. P. Gudadhe, "Load Time and Link Mapping: Enhancing SEO experience for Private University Websites in Maharashtra", *multiensayos*, let. 9, št. 18, str. 21–33, jul. 2023, doi: 10.5377/multiensayos.v9i18.16428.

[29] M. J. Shayegan in M. Kouhzadi, "An Analysis of the Impact of SEO on University Website Ranking", 2020, doi: 10.48550/ARXIV.2009.12417.

[30] SEO Site Checkup, "Search Engine Optimization Made Easy". Pridobljeno: 28. maj 2025. [Na spletu]. Dostopno na: https://seositecheckup.com/

[31] K. I. Roumeliotis in N. D. Tselikas, "A Machine Learning Python-Based Search Engine Optimization Audit Software", *Informatics*, let. 10, št. 3, str. 68, avg. 2023, doi: 10.3390/informatics10030068.

[32] SEO Site Checkup, "Some of our SEO Tools". Pridobljeno: 29. maj 2025. [Na spletu]. Dostopno na: https://seositecheckup.com/tools

[33] Google Search Central, "Google Search Central Blog", Rolling out the mobile-friendly update. Pridobljeno: 10. junij 2025. [Na spletu]. Dostopno na: https://developers.google.com/search/blog/2015/04/rolling-out-mobile-friendly-update

[34] V. Gupta, P. Kaur, in H. Singh, "Bi-Level Decision Tree Approach for Web Quality Assessment", *IEEE Access*, let. 12, str. 130926–130938, 2024, doi: 10.1109/ACCESS.2024.3456808.

[35] C. Tam, F. C. Pereira, in T. Oliveira, "What influences the purchase intention of online travel consumers?", *Tourism and Hospitality Research*, let. 24, št. 2, str. 304–320, 2022, doi: 10.1177/14673584221126468.

[36] T. Stoyanova in K. Anguelov, "Indicators for determining the effective level of digitalization in Higher Education", *JESI*, let. 11, št. 3, str. 246–264, mar. 2024, doi: 10.9770/jesi.2024.11.3(17).

[37] A. Setiawan, Z. Harahap, D. Syamsuar, in Y. N. Kunang, "The Optimization of Website Visibility and Traffic by Implementing Search Engine Optimization (SEO) in Palembang Polytechnic of Tourism", *CommIT (Communication and Information Technology) Journal*, let. 14, št. 1, str. 31, maj 2020, doi: 10.21512/commit.v14i1.5953.

▪

Marko Urh je zaposlen na Univerzi v Mariboru, Fakulteti za organizacijske vede kot višji predavatelj za področje informacijski sistemi in kadrovski management. Njegovo raziskovalno delo je osredotočeno na spletno programiranje, uporabniško izkušnjo, geografskem informacijske sisteme, e-izobraževanje in razvoj človeških virov.

▪

Eva Jereb je profesorica na Katedri za kadrovske in izobraževalne sisteme na Fakulteti za organizacijske vede Univerze v Mariboru. Njena glavna raziskovalna področja so visokošolsko izobraževanje, e-izobraževanje, plagiatorstvo, igrifikacija v izobraževanju, razvoj človeških virov, kadrovski ekspertni sistemi in delo na daljavo.

▪

Alenka Baggia je zaposlena na Univerzi v Mariboru, Fakulteti za organizacijske vede kot docentka za področje informacijski sistemi. Njeno raziskovalno delo je osredotočeno na sprejetje novih tehnologij in vlogo informacijskih sistemov v trajnostnem razvoju. Je članica Laboratorija za kakovost in testiranje programske opreme ter certificirana inštruktorica Oracle Academy.

Obširna evalvacija komercialnih velikih jezikovnih modelov na področju sklepanja v slovenskem jeziku in slovnice

Miha Malenšek, Domen Vreš, Marko Bajec
Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Večna pot 113, 1000 Ljubljana
miha.malensek@fri.uni-lj.si, domen.vres@fri.uni-lj.si, marko.bajec@fri.uni-lj.si

Izvleček

Uporaba velikih jezikovnih modelov (VJM) se hitro širi tudi v slovenskem prostoru, vendar je njihova dejanska zmogljivost za slovenski jezik še vedno slabo sistematično ovrednotena. V tem članku predstavljamo obširno primerjalno evalvacijo najpogosteje uporabljanih komercialnih in odprtih VJM v kontekstu slovenščine. V evalvacijo smo vključili modele štirih večjih ponudnikov (OpenAI, Google, Anthropic in Mistral) ter domači model GaMS-27B-Instruct in jih ovrednotili z raznolikim naborom učnih množic, ki preverjajo sposobnosti sledenja navodilom, razumsko sklepanje, zanesljivost odgovorov, slovnične kompetence ter koherentnost besedila. Uporabili smo prevedene standardizirane primerjalne naloge (npr. ARC, HellaSwag, TruthfulQA, GSM8K), specializirano množico za slovnične napake DASSLE 1.0 ter nabor resničnih pogovorov Slovenske pogovorne arene. Rezultati kažejo, da sodobni komercialni modeli dosegajo visoko uspešnost pri nalogah razumevanja in sklepanja v slovenskem jeziku, zlasti GPT-5.1 z visokim nivojem premišljanja in Gemini-2.5-Pro, medtem ko odprti modeli, kot je Mistral Large 3 kljub omejenim virom dosegajo konkurenčne rezultate. Nasprotno pa evalvacija slovnične kompetence razkriva, da ostaja morfološka in skladenjska kompleksnost slovenskega jezika velik izziv za vse obravnavane modele. Članek s tem nudi celovit vpogled v trenutno stanje zmogljivosti VJM za slovenščino.

Ključne besede: veliki jezikovni modeli (VJM), evalvacija, analiza slovničnih napak, obdelava naravnega jezika

Comprehensive Evaluation of Commercial Large Language Models for Reasoning in Slovenian Language and Grammar

Abstract

The use of large language models (LLMs) is rapidly expanding in the Slovenian context; however, their actual performance on the Slovenian language remains insufficiently and unsystematically evaluated. In this paper, we present an extensive comparative evaluation of the most widely used commercial and open-source LLMs with respect to Slovenian. The evaluation includes models from four major providers (OpenAI, Google, Anthropic, and Mistral), as well as the domestic models GaMS-27B-Instruct and GaMS3-12B-Instruct, and assesses them using a diverse set of benchmarks targeting instruction following, reasoning abilities, answer reliability, grammatical competence, and textual coherence. We employ translated standardized benchmark tasks (e.g., ARC, HellaSwag, TruthfulQA, GSM8K), the specialized grammatical error dataset DASSLE 1.0, and a collection of real-world conversations from the Slovenian Conversational Arena. The results show that contemporary commercial models achieve high performance on comprehension and reasoning tasks in Slovenian—most notably GPT-5.1 with a high level of deliberative reasoning and Gemini-2.5-Pro—while open models such as Mistral Large 3 attain competitive results despite more limited resources. In contrast, the evaluation of grammatical competence reveals that the morphological and syntactic complexity of Slovenian remains a significant challenge for all evaluated models. Overall, the paper provides a comprehensive overview of the current state of LLM performance for the Slovenian language.

Keywords: Large Language Models (LLM), Evaluation, Grammatical Error Analysis, Natural Language Processing

1 UVOD

Uporaba velikih jezikovnih modelov (VJM) postaja vse bolj pogosta na vseh področjih življenja, nastop generativnih jezikovnih modelov v obliki splošno sposobnih, pogovornih storitev, kot so ChatGPT, Gemini, Claude in drugi, pa so uporabo še povišali. Iz letnih raziskav podjetja McKinsey in univerze Stanford je razvidno, da je globalna uporaba generativnih orodij narasla s 55 % v letu 2023, na 88 % v letu 2025 [1] [2]. V Sloveniji anketa Centra za družboslovno informatiko razkriva podoben trend – delež redne uporabe generativnih orodij se je skoraj podvojil z 25 % v letu 2024 na 48 % uporabnikov v letu 2025 [3]. Primerjave nakazujejo, da Slovenija za najrazvitejšimi državami zaostaja za približno eno leto, kar pomeni, da se bo delež uporabe verjetno še povečal.

Med generativnimi orodji najbolj prevladujejo veliki jezikovni modeli, od katerih v Sloveniji prevladuje ChatGPT, ki ga uporablja 83 % uporabnikov generativnih orodij. Hiter porast uporabe VJM spremlja tudi povečanje raznovrstnosti modelov, dostopnih s komercialnimi, plačljivimi programskimi vmesniki (API). V zadnjih letih so vodilni ponudniki OpenAI, Google, Anthropic in Mistral redno posodabljali ponujene VJM ter uporabnikom omogočili dostope do zmogljivejših in zanesljivejših storitev.

Kljub temu pa se v praksi izkaže, da se zmogljivosti posameznih modelov močno razlikujejo, še posebej pri zahtevnejših nalogah v slabše zastopanih jezikih, kot je slovenščina. Ob hitri adopciji v vsakodnevni rabi se tako pojavi ključno vprašanje o generalni sposobnosti najpogosteje uporabljanih VJM v kontekstu slovenskega jezika.

V tem članku zato predstavljamo obširno evalvacijo trenutno najpogosteje uporabljanih VJM. Vključujemo modele iz vseh štirih večjih ekosistemov (OpenAI, Google Gemini, Anthropic Claude in Mistral) in jih ovrednotimo na prevedenih, standardiziranih primerjalnih testih, ki se pogosto uporabljajo za ovrednotenje sposobnosti VJM v angleškem jeziku, kakor tudi na pogovorni kakovosti in skladnosti odgovorov ter slovničnih sposobnostih.

2 IZBRANI PONUDNIKI

Za evalvacijo smo izbrali storitve podjetij Google, OpenAI, Anthropic in Mistral ter evalvirali širok nabor ponujenih modelov različnih cenovnih nivojev.

Tabela 1: **Nabor evalviranih modelov. Cena podan v dveh vrednostih, na milijon vhodnih in izhodnih tokenov. Pri tem je potrebno opozoriti, da razmišljujoči (angl. reasoning) modeli, štejejo tokene razmišljanja v izhodne tokene.**

Podjetje	Model	Cena (1M vhodnih / 1M izhodnih tokenov)
Google	gemini-2.5-pro	$2.50 / $15.00
	gemini-2.5-flash	$0.30 / $2.50
OpenAI	gpt-5.1	$1.25 / $10.00
	gpt-5	$1.25 / $10.00
	gpt-5-mini	$1.25 / $2.00
	gpt-5-nano	$0.05 / $0.40
	gpt-4.1	$2.00 / $8.00
	gpt-4o-mini	$0.15 / $0.60
Anthropic	Claude Opus 4.1	$15.00 / $75.00
	Claude Sonnet 4.5	$3.00 / $15.00
	Claude Haiku 4.5	$1.00 / $5.00
Mistral	Mistral Large 3	$0.50 /$1.50
	Mistral Medium 3.1	$0.40 / $2.00
	Mistral Small 3.2	$0.10 / $0.30
FRI	GaMS-27B-Instruct	Lokalna postavitev

Modele in cene smo navedli v Tabeli 1. Cene so podane v standardnem formatu, ki ga uporabljajo vsi štirje ponudniki, ki ceno uporabe sestavi iz količine vhodnih in izhodnih tokenov (t.j. besednih enot, ki jih model generira). Podana je kot cena na milijon vhodnih in izhodnih tokenov, izpisana iz dokumentacije API vmesnikov v času evalviranja modelov.

Vsi štirje ponudniki imajo podoben API vmesnik, zato med posamezno implementacijo ni dosti razlik. Pri uporabi je potrebno paziti na omejitve količine klicev na minuto ter na samo porabo zakupljenega dobroimetja − vsi štirje ponudniki namreč dostopa do API funkcionalnosti ne omejijo ali ustavijo po porabi celotnega zakupljenega dobroimetja, omogočajo le možnosti obveščanja, ko poraba preseže definirano vrednost.

3 UČNE MNOŽICE

Za podrobno evalvacijo sposobnosti velikih jezikovnih modelov (VLM) v slovenskem jeziku smo izbrali tri učne množice, ki z raznolikimi nalogami v slovenskem jeziku preverjajo zmožnosti sledenja navodilom, sposobnosti razumskega sklepanja, ekstrakcije in uporabe informacij iz danega konteksta, nagnjenosti k zavajanju oz. podajanju neresničnih ali napačnih informacij, slovnično in pravopisno kompetenco ter pogovorne sposobnosti.

Za te namene smo uporabili prostodostopni množici Slovenian LLM Evaluation[1] objavljeno na repozitoriju HuggingFace in množico DASSLE 1.0[2], objavljeno na repozitoriju Clarin, ter označene sledi pogovorov Slovenske pogovorne arene[3].

Množica Slovenian LLM Evaluation je skupek specifičnih nalog, s katerimi evalviramo generalne sposobnosti VJM in sposobnosti sledenja kompleksnih navodil in uporabo konteksta ter izluščenih informacij, podanih v slovenskem jeziku. Z množico DASSLE 1.0 evalviramo slovnične in oblikoslovne sposobnosti VJM. Slovenščina je morfološko kompleksen jezik, zato je evalvacija na tej množici še posebej pomembna. Množica pogovorov slovenske pogovorne arene evalvira tok pogovora z VJM ter reakcije le-teh na včasih nesmiselne vhode.

3.1 DASSLE 1.0

Množica DASSLE 1.0 (Dataset of Authentic and Synthetic Slovene Language Errors) je sestavljena iz 7385 ročno pripravljenih primerov, ki vsebujejo poved z edinstveno slovnično ali pravopisno napako, pravilno popravljeno poved ter makro- in mikro-kategorijo napake. Z množico DASSLE preverjamo dve pomembni sposobnosti VLM:

- ali je model sposoben **pravilno identificirati makro-kategorijo napake** in
- ali je model sposoben napako pravilno odpraviti ter predlagati pravilen popravek.

Evalvacija na tej množici nam da pomemben vpogled v **jezikovno, slovnično in pravopisno kompetenco** VLM, kar je še posebej pomembno v kontekstu slovenskega jezika.

3.2 Pogovori Slovenske pogovorne arene

Množica vsebuje primere ročno ustvarjenih, uporabniških pogovorov z VLM na Slovenski pogovorni areni. Z uporabo uporabniških pozivov simuliramo interakcijo med VLM in človeškim uporabnikom, nato pa z uporabo panele sodnikov (LLM-as-a-judge) presodimo pogovorno smiselnost in koherentnost celotne sledi pogovora.

Evalvacija na tej množici nam da vpogled v pogovorne sposobnosti VLM na raznolikih (in včasih tudi nesmiselnih) uporabniških pozivov.

3.3 Slovenian LLM Evaluation

Množica slovenian-llm-eval je prostodostopna množica na repozitoriju HuggingFace, ki vključuje skupek nalog, ki se pogosto uporabljajo za evalvacijo VLM. Naloge so strojno prevedene v slovenski jezik in ročno pregledane ter zaobjemajo predvsem sposobnosti sledenju daljših navodil, razumevanju konceptov ter ekstrakciji in uporabi relevantne informacije v novem kontekstu.

3.3.1 ARC (AI2 Reasoning Challenge)

ARC-Challenge in *ARC-Easy* [4] sta množici vprašanj, zasnovani za preverjanje, ali lahko veliki jezikovni modeli pravilno odgovarjajo na naravoslovna vprašanja iz osnovnošolskih preverjanj znanj v ZDA. Naloge ne temeljijo zgolj na preprostem iskanju ključnih besed ali pomnjenju dejstev, temveč zahtevajo razumevanje, sklepanje in uporabo znanstvenega znanja. Naloge so ločene na lažja vprašanja, na katera je mogoče pogosto odgovoriti z osnovnim znanjem, ter na modelom težja vprašanja, saj zahtevajo povezovanje dejstev, razume-

[1] https://huggingface.co/datasets/cjvt/slovenian-llm-eval
[2] https://vlo.clarin.eu/record/https_58__47__47_hdl.handle.net_47_11356_47_2052_64_format_61_cmdi
[3] https://arena.cjvt.si/sl

vanje znanstvenih konceptov, predvsem pa prepoznavanje vzrokov in posledic ter logičnih razmerij. Naloga tako preverja, ali je model sposoben preseči preprosto pomnjenje in priklic informacij.

Posamezna naloga je sestavljena iz vprašanja in seznama možnih odgovorov med katerimi model izbira.

3.3.2 HellaSwag

HellaSwag [5] je zbirka nalog, zasnovana za preverjanje logičnega sklepanja velikih jezikovnih modelov. Posamezna naloga vključuje kratke, nedokončane opise situacij, ki jim sledi več možnih nadaljevanj. Model mora izbrati najbolj smiselno nadaljevanje danega opisa. Vse podane možnosti so slovnično pravilne in slogovno naravne, pravilna možnost pa je tista, ki se ujema z realističnim potekom dogodkov. Naloga zato ne preverja le površinskega razumevanja jezika, temveč predvsem razumevanje širšega konteksta in implikacij le-tega.

3.3.3 TruthfulQA

TruthfulQA [6] je zbirka nalog, namenjena preverjanju zanesljivosti in resnicoljubnosti odgovorov velikih jezikovnih modelov in njihovi odpornosti na zavajajoče odgovore, še posebej v primerih, ko se vprašanja opirajo na napačne predstave, teorije zarote, zavajajoče trditve ali pogosto ponavljane netočnosti z interneta. Cilj ni le preverjanje prepoznavanja resničnih dejstev, temveč tudi sposobnosti modela, da prepozna in zavrne napačne premise. Zbirka vključuje tako vprašanja z enim pravilnim odgovorom *(truthfulqa_mc1)* in vprašanja z več pravilnimi odgovori *(truthfulqa_mc2)*. Pri slednjih v tabeli rezultatov predstavimo tako strogo natančnost *(model pravilno izbere vse možne odgovore)* in povprečno natančnost *(koliko pravilnih vprašanj je model izbral)*.

3.3.4 BoolQ

BoolQ [7] je zbirka nalog, ki preverja sposobnost velikih jezikovnih modelov, da berejo, razumejo in presojajo informacije v danem besedilu. Vsaka naloga vsebuje vprašanje in kratek odlomek iz Wikipedije, na podlagi katerega mora model na vprašanje odgovoriti z *da* ali *ne*. Gre za naravna vprašanja, zbrana iz spletnih iskanj, kar pomeni, da so pogosto nepopolna, pogovorna ali implicitno zastavljena, zato pravilni odgovor zahteva razumevanje konteksta, prepoznavo parafraz, logičnih povezav in prikritih negacij. S tem preveri sposobnost sklepanja na podlagi danega konteksta in razločevanje med eksplicitno in implicitno podanim znanjem.

3.3.5 OpenBookQA

OpenBookQA [8] je zbirka nalog, zasnovana za preverjanje sposobnosti velikih jezikovnih modelov, da kombinirajo osnovno znanstveno znanje z vsakdanjim sklepanjem. Naloga temelji na konceptu izpita »odprte knjige«, v kateri ima model poleg vsakega vprašanja na voljo še zbirko približno 1300 osnovnih dejstev, ki predstavljajo temeljno znanje s področij, kot so fizika, biologija in kemija. Vprašanja so oblikovana tako, da zgolj prepoznavanje teh dejstev ni dovolj, model mora znanje pravilno uporabiti v novi situaciji. Naloge vključujejo več odgovorov, od katerih je zgolj en pravilen, ostali pa so pogosto zavajajoči. S tem preverimo sposobnost povezovanja informacij, razumevanje naravnih pojavov in logično sklepanje.

3.3.6 WinoGrande

WinoGrande [9] je zbirka nalog, namenjena preverjanju sposobnosti sklepanja in razumevanja konteksta jezikovnih modelov, pri čemer se osredotoča na izziv referenčne razjasnitve *(angl. coreference resolution)*. Vsak primer vsebuje stavek z manjkajočim ali dvoumnim delom, ki ga je treba zapolniti z eno izmed dveh ponujenih možnosti. Model mora na podlagi zdrave pameti, razumevanja vzorčnih odnosov, fizičnega in socialnega znanja ter logičnega sklepanja izbrati tisto možnost, ki najbolj smiselno dopolni poved. Obe možnosti sta slovnično in slogovno pravilni, pravilna rešitev pa je tista, ki smiselno odraža pravilen potek dogodkov. Primeri so za modele zahtevni in predstavljajo test globokega razumevanja tako jezika, kot širšega konteksta.

3.3.7 PIQA (Physical Interaction Question Answering)

PIQA [10] je zbirka podatkov, zasnovana za preverjanje praktičnega sklepanja velikih jezikovnih modelov. Vsak primer vsebuje opis enostavnega cilja ali opravila iz vsakdanjega življenja ter dve možni rešitvi oziroma opisa poti do tega cilja. Model mora izbrati tisto rešitev, ki je fizično izvedljiva in smiselna v resničnem svetu. Naloga se osredotoča na razumevanje lastnosti predmetov, naravnih zakonov, vzročnih razmerij in običajne rabe orodij in materialov. Naloga s tem preverja, ali modeli premorejo

praktično, fizično znanje vsakdanjega sveta in ali ga znajo uporabiti v dani situaciji.

3.3.8 GSM8K (Grade School Math 8K)

GSM8K [11] je zbirka besedilnih nalog s področja matematike. Naloge preverjajo aritmetične sposobnosti in logično sklepanje z več koraki. Naloge zahtevajo pravilno interpretacijo podatkov ter sistematično izvedbo računskih korakov, ki lahko vključujejo seštevanje, odštevanje, deljenje in množenje ter delo z enotami in preprostimi enačbami. Ključni izziv je razčleniti besedilo naloge, izluščiti relevantne informacije in jih uporabiti za pravilno rešitev, ki pogosto zahteva več korakov. Naloge ocenjujejo zanesljivosti matematičnega sklepanja in doslednost modelov z ozirom na pomnjenje relevantnih informacij in jasno razkrijejo, ali model razume problem ter zna metodično sklepati in računati.

Evalvacija VLM na tem naboru nalog nam omogoči osnovni vpogled v sposobnosti VLM v sledenju navodil in uporabi informacij podanih v slovenskem jeziku na specifičnih nalogah. Izmed izbranih nalog gre za najlažje, saj ne preverja kompetenc v pisanju slovenskega jezika, temveč le v sposobnosti razumevanja in povezovanja informacij podanih v slovenskem jeziku.

4 REZULTATI

Za pridobivanje rezultatov evalvacije smo za vsako učno množico pripravili ogrodje, s katerim lahko definiramo nabor modelov in podnalog, na katerih se opravlja evalvacija. Ker se uporaba komercialnih vmesnikov med ponudniki nekoliko razlikuje, so trenutno podprti samo modeli izbranih ponudnikov, predstavljenih v Tabeli 1, ter modeli kompatibilni s knjižnico vLLM[4]. Razen razlike v samem postopku klica storitve se struktura uporabljenih pozivov, struktura pričakovanega odgovora, ter ocenjevalna skripta med modeli ne razlikuje. Rezultati vsake evalvacije se za potrebe pregledov in ocenjevanja shranijo v modelu in v nalogi edinstveno datoteko.

Za vsako nalogo smo definirali predlogo poziva, ki definira vhod in pričakovani izhod. Evalvacija je temeljila na »one-shot[5]« pristopu, kjer se modelu poda problem, ki ga mora rešiti, in en vzorčni primer rešitve. Slednje zagotovi, da se modeli držijo pričakovane strukture odgovora, kar olajša ocenjevanje odgovora. S tem smo preverili iztočno sposobnost modelov – ostalih pristopov, ki bi izboljšali delovanje generativnih modelov, kot na primer »fine-tuning[6]«, nismo uporabljali, saj nas je zanimala predvsem splošna sposobnost modelov v slovenskem jeziku.

Nekateri moderni modeli za odgovore uporabljajo pristop »premišljanja[7]«, ki modelu omogoča veriženja misli na podlagi uporabnikove zahteve. Pristop zanesljivo izboljša kvaliteto odgovorov modelov [12], zato ga komercialni ponudniki privzeto implementirajo v lastne modele. V primeru modela GPT-5.1 podjetja OpenAI smo lahko evalvirali tako visok nivo premišljanja in nizek nivo (brez) premišljanja, saj njihov API omogoča določitev nivoja le-tega. Pri tem je potrebno poudariti, da se s tem tudi bistveno poviša nivo izhodnih enot, kar pa vpliva na ceno.

Rezultati evalvacij na posameznih izbranih učnih množicah so predstavljeni v Tabelah 2 in 3. Za preglednost tabel ne vključujemo cene, ki je že prikazana v Tabeli 1.

V rezultatih predvsem izstopajo modeli GPT-5.1 z visokim nivojem premišljevanja, Gemini 2.5 Pro in odprti model Mistral Large 3. Vsi trije so se dobro izkazali v reševanju nalog Slovenian LLM Evaluation (Tabela 2), kjer so pokazali dobre sposobnosti ekstrakcije in uporabe informacij iz navodil, podanih v slovenskem jeziku. Pri tem je potrebno poudariti, da gre za zelo specifične naloge, pogosto z vnaprej definiranim naborom možnih rešitev, ki pa vendarle zahtevajo razumevanje, ekstrakcijo in uporabo podanih informacij.

Za referenco podajamo tudi rezultate modela GaMS-27B-Instruct, VJM pripravljenega na Fakulteti za računalništvo in informatiko, predučenega na slovenskem jeziku. V primerjavi s komercialnimi modeli je razlika velika, saj model omejuje dostopnost in kvaliteta virov, število parametrov, čas predučenja ter prilagajanje slovenskemu jeziku, kvaliteta inštrukcijske množice in nenazadnje tudi dostop do računskih virov ter finančno breme obsega predučenja.

Medtem ko so rezultati na množici nalog Slovenian LLM Evaluation (Tabela 2) zelo obetavni, rezultati evalvacije na množici DASSLE 1.0 (Tabela 3)

[4] https://docs.vllm.ai/en/latest/
[5] angl., pristop enojnega strela
[6] angl., fino učenje, učenje modela na specifični nalogi
[7] angl., reasoning

kažejo drugačno sliko. Izkaže se, da so slovnične in oblikoslovne lastnosti slovenskega jezika še vedno velik zalogaj za VJM. Tudi najuspešnejši modeli na nalogah Slovenian LLM Evaluation se težko spopadajo tako z razpoznavo kategorije napake kot z odpravljanjem le-te.

Uporabili smo strogo metriko natančnosti popravka, kjer smo za pravilni odgovor šteli le niz, ki se je popolnoma skladal s popravljeno povedjo podano v množici, saj ima večina primerov enolično rešitev[8]. Rezultati kažejo, da se modeli dobro izkažejo le pri preprostejših napakah črkovanja in zapisa, kjer gre predvsem za popravke začetnic, pisanje skupaj ali narazen in zapis glasu ali glasovnega sklopa v besedi. V težjih kategorijah, kot so besedišče, oblikoslovje in predvsem skladnja pa se rezultati bistveno poslabšajo. To nam da vedeti, da se sicer modeli zavedajo preprostejših napak, a globljega razumevanja nians slovenskega jezika kljub temu primanjkuje. Še posebej zaskrbljujoči so rezultati pri napakah iz besedišča, kar lahko močno vpliva na razumevanje kompleksnejših navodil in posledično na kvaliteto danih odgovorov.

Rezultate pogovorne množice smo prikazali v Tabeli 4, kjer prikazujemo število zmag, porazov, izenačenj ter oceno ELO. Rezultati so pridobljeni s pristopom panele VJM sodnikov, ki glasujejo o primerjalni kvaliteti odgovorov dveh modelov. V panelo so vključeni vsi modeli, vključeni v evalvacijo, zaradi skrbi pristranskosti pa smo jim onemogočili samoocenjevanje. Zmage in izenačenja se agregirajo v direktno oceno, ki predstavlja seštevek števila zmag (1 točka) in izenačenega rezultata (0.5 točke) ter v šahovski ELO, kjer gre za relativno oceno sposobnosti VJM. Za vpogled v složnost ocen modelov sodnikov smo izračunali Kappa statistike in povprečen delež strinjanja, prikazano v Tabeli 5. Statistike složnosti kažejo na zanesljiv nivo strinjanja med VJM sodniki, kar doprinaša k verodostojnosti rezultatov.

Rezultati sami so skladni z doseženimi rezultati na evalvacijah na množicah Slovenian LLM Evaluation in DASSLE 1.0, saj se najboljša modela (Gemini-2.5-pro in GPT-5.1) pojavita na vrhu, z najvišjim številom zmag in najvišjo oceno ELO. Mistral Me-

Rezultati: Slovenian LLM Evaluation Dataset

Tabela 2: **Rezultati evalvacije na množici slovenian-llm-eval. Novi gpt-5.1 modeli podjetja OpenAI omogočajo nastavitev ‚reasoning' parametra. Rezultati vključujejo zmogljivosti v načinu ‚high' in ‚none'. Z zeleno so označeni najboljši rezultati glede nanalogo, z rdečo pa najslabši.**

Task / Model	gpt-5.1 high reasoning	gpt-5.1 no reasoning	gpt-5	gpt-4o-mini	gpt-5-mini-2025-08-07	gpt-4.1-2025-04-14	gpt-5-nano-2025-08-07	gemini-2.5-pro	gemini-2.5-flash	Mistral Large 3	Mistral Medium 3.1	Mistral Small 3.2	Claude Opus 4.1	Claude Sonnet 4.5	Claude Haiku 4.5	GaMS-27B-Instruct
arc_challenge	0.938	0.918	0.944	0.848	0.94	0.93	0.918	0.964	0.946	0.898	0.896	0.836	0.934	0.912	0.894	0.543
arc_easy	0.986	0.97	0.983	0.94	0.982	0.976	0.972	0.986	0.982	0.97	0.936	0.93	0.978	0.982	0.968	0.773
hellaswag	0.892	0.858	0.916	0.794	0.852	0.88	0.726	0.92	0.89	0.83	0.822	0.784	0.938	0.952	0.84	0.723
truthfulqa_mc1	0.832	0.802	0.813	0.684	0.788	0.794	0.718	0.854	0.744	0.67	0.658	0.586	0.898	0.938	0.78	0.458
truthfulqa_mc2	0.416	0.406	0.419	0.328	0.378	0.504	0.25	0.586	0.51	0.35	0.22	0.312	0.536	0.414	0.438	0.276
	0.675	0.676	0.695	0.697	0.661	0.741	0.718	0.804	0.751	0.675	0.714	0.724	0.845	0.885	0.734	0.878
Boolq	0.868	0.88	0.879	0.834	0.882	0.87	0.858	0.874	0.882	0.846	0.852	0.846	0.908	0.906	0.842	0.862
openbookqa	0.936	0.906	0.936	0.824	0.922	0.91	0.894	0.918	0.93	0.84	0.812	0.772	0.86	0.872	0.842	0.322
winogrande	0.842	0.714	0.865	0.616	0.82	0.782	0.682	0.836	0.838	0.628	0.865	0.588	0.772	0.788	0.64	0.696
Piqa	0.938	0.91	0.922	0.854	0.9	0.902	0.64	0.928	0.912	0.82	0.852	0.752	0.91	0.86	0.836	0.749

[8] https://wiki.cjvt.si/books/11-developmental-corpus-solar/page/annotation-guidelines

Rezultati: DASSLE 1.0

Tabela 3: **Tabela 3: Rezultati evalvacije na množici DASSLE 1.0. Rezultati predstavljajo strogo natančnost klasifikacije kategorije in predlaganega popravka. Prikazuje tudi natančnost popravkov znotraj posamezne kategorije. Z zeleno so označeni najboljši rezultati glede na nalogo (vrstica), z rdečo pa najslabši.**

Task / Model	gemini-2.5-pro	gemini-2.5-flash	gpt-5.1 high reasioning	gpt-5.1 no reasioning	gpt-5	gpt-5-mini	gpt-5-nano	gpt-4.1	gpt-4o-mini	Mistral Large 3	Mistral Medium 3.1	Mistral Small 3.2	Claude Opus 4.1	Claude Sonnet 4.5	Claude Haiku 4.5	GaMS-27B-Instruct
strict accuracy	0.546	0,463	0,555	0,489	0,579	0,45	0,31	0,52	0,368	0,392	0,312	0,257	0,516	0,449	0,348	0,382
category accuracy	0.54	0,501	0,607	0,527	0,601	0,478	0,334	0,542	0,394	0,4	0,374	0,23	0,548	0,521	0,426	0,322
Besedišče	0.48	0,39	0,46	0,41	0,54	0,22	0,13	0,44	0,15	0,28	0,2	0,13	0,37	0,23	0,19	0,24
Oblikoslovje	0.51	0,36	0,54	0,45	0,55	0,47	0,26	0,5	0,36	0,39	0,31	0,2	0,51	0,46	0,36	0,37
Skladnja	0.25	0,27	0,22	0,21	0,23	0,19	0,1	0,2	0,11	0,16	0,16	0,07	0,24	0,16	0,11	0,1
Zapis	0.75	0,66	0,77	0,73	0,77	0,69	0,53	0,71	0,58	0,56	0,45	0,45	0,7	0,68	0,48	0,54
Črkovanje	0.74	0,64	0,79	0,65	0,8	0,68	0,53	0,75	0,64	0,57	0,44	0,43	0,76	0,71	0,6	0,66

Rezultati: Sledi pogovorov Slovenske pogovorne arene

Tabela 4: **Ocene sledi pogovorov, pridobljene s panelo VJM sodnikov. Ocena je izračunana iz utežene vsote zmag in izenačenj, medtem ko je ELO relativna ocena sposobnosti. ELO bolje oceni modele, ki nepričakovano premagajo boljše ocenjene modele.**

Model	Zmage	Porazi	Izenačenja	Ocena	ELO
gemini-2.5-pro	202	37	11	207.5	1916.7
gpt-5.1-2025-11-13	178	57	15	185.5	1836.3
mistral-medium-2508	171	60	19	180.5	1713.8
gpt-5-2025-08-08	154	79	17	162.5	1816.6
gemini-2.5-flash	137	97	16	145	1597.7
claude-sonnet-4-5	116	121	13	122.5	1445.7
claude-opus-4-1	115	127	8	119	1410.2
gpt-4.1-2025-04-14	108	133	9	112.5	1423.8
claude-haiku4-5	84	160	6	87	1262.1
gpt-4o-mini-2024-07-18	31	214	5	33.5	1151.2
mistral-small-2506	18	229	3	19.5	925.8

Tabela 5: **Metrike strinjanja med VJM sodniki. Odstotek parovnega strinjanja je direkten izračun, kolikokrat so se modeli za iste sledi pogovorov strinjali. Cohenova kappa ima nabor vrednosti med -1 in 1, kjer 0 pomeni naključna ujemanja, 1 pa popolno ujemanje. Fleissova kappa je razširjena Cohenova kappa za več kot dva ocenjevalca.**

Odstotek parnega strinjanja	77.5%
Povprečna Cohenova kappa	0.578
Fleissova kappa	0.5

dium po številu zmag doseže tretje mesto, a ga ELO postavi na četrto z modelom GPT-5 na tretjem, saj ocena ELO ocenjuje relativno sposobnost modela (t.j. zmago proti predvideno slabšem modelu oceni slabše, kot presenetljivo zmago proti boljšem modelu). Tudi to se sklada z rezultati na Slovenian LLM Evalutaion in DASSLE 1.0 – medtem ko je Mistral Medium generalno sposoben model, je GPT 5 v neka-

terih kategorijah boljši tudi od GPT 5.1, kar pomeni, da je verjetno v določenih pogovorih podajal boljše odgovore kot najvišje ocenjeni modeli.

5 ZAKLJUČEK

V tem članku smo predstavili celovito evalvacijo najpogosteje uporabljanih velikih jezikovnih modelov v kontekstu slovenskega jezika. Z vključitvijo modelov iz štirih večjih komercialnih ekosistemov ter enega domačega, odprtega modela smo predstavili vpogled v trenutno stanje splošnih, jezikovnih in pogovornih sposobnosti VJM v slovenskem jeziku. Evalvacijo smo izvedli na raznolikem naboru učnih množic, ki skupaj pokrivajo sledenje navodilom, razumsko sklepanje, zanesljivost odgovorov, slovnično kompetenco ter besedilno koherentnost.

Rezultati kažejo, da so sodobni komercialni modeli, predvsem GPT-5.1 z visokim nivojem premišljanja ter Gemini-2.5-Pro, dosegli visoko raven uspešnosti pri nalogah, ki preverjajo razumevanje in uporabo informacij v slovenskem jeziku, zlasti na množici Slovenian LLM Evaluation. Ti modeli so se izkazali kot zanesljivi pri reševanju strukturiranih nalog, ki temeljijo na razumevanju navodil in logičnem sklepanju, kar potrjuje njihovo uporabnost tudi v slovenskem jezikovnem prostoru. Odprti model Mistral Large 3 se je kljub bistveno nižji ceni in odprti naravi presenetljivo dobro kosal z najboljšimi komercialnimi modeli, kar kaže na hiter napredek odprtokodnih pristopov.

Po drugi strani evalvacija na množici DASSLE 1.0 razkriva pomembne omejitve vseh obravnavanih modelov. Oblikoslovna in predvsem skladenjska kompleksnost slovenskega jezika ostaja velik izziv tudi za najzmogljivejše VJM. Modeli se relativno dobro spopadajo s preprostimi pravopisnimi napakami, vendar njihova uspešnost hitro upade pri globljih jezikovnih pojavih, kot so besediščne, oblikoslovne in skladenjske napake. To nakazuje, da trenutni modeli še nimajo zadostno robustnega razumevanja slovenskega jezika, kar lahko omejuje njihovo zanesljivost pri zahtevnejših nalogah, kjer je natančna raba jezika ključna.

Rezultati evalvacije pogovornih sposobnosti dodatno potrjujejo ugotovitve iz preostalih nalog. Najbolje ocenjeni modeli po ELO lestvici so hkrati dosegali visoke rezultate tudi na drugih učnih množicah, kar kaže na konsistentnost njihovih splošnih sposobnosti. Uporaba več VJM kot sodnikov se je izkazala za ustrezno, saj statistike složnosti kažejo na zadovoljivo stopnjo strinjanja in s tem na verodostojnost primerjalnih ocen.

Skupno gledano rezultati kažejo, da so veliki jezikovni modeli dovolj zreli za široko uporabo v slovenskem jeziku pri nalogah razumevanja, sklepanja in splošne pogovorne interakcije. Kljub temu pa ostajajo izrazite vrzeli pri globlji jezikovni pravilnosti, kar je še posebej pomembno za rabo v izobraževanju, javni upravi, pravu in drugih domenah, kjer je natančnost jezika ključnega pomena.

LITERATURA

[1] A. Singla, A. Sukharevsky, M. Chui in B. Hall, "The state of AI," McKinsey & Company, 2025.

[2] N. Maslej, L. Fattorini, R. Perrault, Y. Gil, V. Parli, N. Kariuki, E. Capstick, A. Reuel, E. Brynjolfsson, J. Etchemendy, K. Ligett, T. Lyons, J. Manyika, J. C. Niebles, Y. Shoham, R. Wald, T. Walsh, A. Hamrah, L. Santarlasci, J. B. Lotufo, A. Rome, A. Shi in S. Oak, "Artificial Intelligence Index Report 2025," Human-Centered Artificial Intelligence, Stanford University, Stanford, 2025.

[3] A. Praček in V. Vehovar, "Umetna inteligenca v Sloveniji 2025/I: Uporabniki GenUI," Center za družboslovno informatiko, Fakulteta za družbene vede, Ljubljana, 2025.

[4] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal in O. Schoenick, "Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge," *arXiv preprint*, p. arXiv: 1803.0547, 2018.

[5] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi in Y. Choi, "HellaSwag: Can a Machine Really Finish Your Sentence?," *arXiv preprint*, p. arXiv:1905.07830, 2019.

[6] S. Lin, J. Hilton in O. Evans, "TruthfulQA: Measuring How Models Mimic Human Falsehoods," *arXiv preprint*, p. arXiv:2109.07958, 2022.

[7] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins in K. Toutanova, "BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions," *arXiv preprint*, p. aXiv:1905.10044, 2019.

[8] T. Mihaylov, P. Clark, T. Khot in A. Sabharwal, "Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering," *arXiv preprint*, p. arXiv:1809.02789, 2018.

[9] K. Sakaguchi, R. Le Bras, C. Bhagavatula in Y. Choi, "WinoGrande: an adversarial winograd schema challenge at scale," *Communications of the ACM*, pp. 99-106, 2021.

[10] Y. Bisk, R. Zellers, R. Le Bras, J. Gao in Y. Choi, "PIQA: Reasoning about Physical Commonsense in Natural Language," *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 7432-7439, 2020.

[11] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse in J. Schulman, "Training Verifiers to Solve Math Word Problems," *arXiv preprint*, p. arXiv:2110.14168, 2021.

[12] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le in D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," *Advances in neural information processing systems*, 35, pp. 24824-24837, 2022.

[13] A. Praček.

▪

Miha Malenšek je raziskovalec in doktorski študent na Fakulteti za računalništvo in informatiko, Univerze v Ljubljani, zaposlen v Laboratoriju za podatkovne tehnologije. V svojem delu se ukvarja predvsem s podpornimi sistemi za varno in sledljivo uporabo VJM v domenah, kjer je zanesljiva in preverljiva uporaba VJM ključnega pomena.

▪

Domen Vreš je raziskovalec in doktorski študent na Fakulteti za računalništvo in informatiko, Univerze v Ljubljani, zaposlen v Laboratoriju za strojno učenje in jezikovne tehnologije. V svojem delu se ukvarja predvsem z učenjem VJM za slovenski jezik, GaMS (Generativni Model Slovenščine).

▪

Marko Bajec je redni profesor na Fakulteti za računalništvo in informatiko Univerze v Ljubljani ter vodja Laboratorija za podatkovne tehnologije in IoT Demo Centra. Predava več predmetov s področja informatike in podatkovnih baz. V okviru aplikativnega in raziskovalnega dela se ukvarja z obvladovanjem informatike ter uporabo podatkovnih tehnologij v okviru različnih domen, kot so internet stvari, pametna mesta, pametni domovi, oskrbovana stanovanja, telemedicina ipd.

Analiza delovanja kopice in napadov dvojne sprostitve

Domen Breznik, Mark Novak, Matevž Pesek
Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Večna pot 113, 1000 Ljubljana
db00709@student.uni-lj.si, mn44954@student.uni-lj.si, matevz.pesek@fri.uni-lj.si

Izvleček

Članek obravnava varnostne ranljivosti, ki nastanejo zaradi nepravilnega upravljanja s pomnilnikom v programih, napisanih v jezikih C/C++. Posvečamo se napadu dvojne sprostitve (angl. double free), ki omogoča napadalcu prevzem nadzora nad pomnilniškim prostorom in potencialno krajo administratorskih privilegijev. Predstavimo pregled sorodnih ranljivosti, kot so prekoračitev kopice (angl. heap overflow) in uporaba po sprostitvi (angl. use-after-free) ter obravnavamo obrambo pred takšnimi napadi. Na osnovi prilagojene aplikacije demonstriramo praktičen napad, prikažemo proces izkoriščanja ranljivosti in analiziramo posledice, ki segajo od nestabilnosti sistema do resnih varnostnih groženj. Članek obravnava obstoječe rešitve, kot so uporaba jezikov z samodejnim upravljanjem s pomnilnikom, alternativnih implementacij funkcije malloc, statične analize kode ter pristopov defenzivnega programiranja.

Ključne besede: dvojna sprostitev, napad, kopica, upravljanje pomnilnika, koši, arene

Analysis of Heap Operation and Double Free Attacks

Abstract

The article examines security vulnerabilities arising from improper memory management in programs written in C/C++. We focus on the double free attack, which enables an attacker to take control of memory space and potentially obtain administrative privileges. A review of related vulnerabilities, such as heap overflow and use-after-free, is presented, along with defenses against such attacks. Based on a customized application, we demonstrate a practical attack, show the process of exploiting the vulnerability, and analyze the consequences, which range from system instability to severe security compromises. The article discusses existing solutions, such as the use of memory-safe programming languages, alternative implementations of the malloc function, static code analysis, and defensive programming approaches.

Keywords: double free, attack, heap, memory management, bins, arenas

1 SEZNAM UPORABLJENIH KRATIC

Kratica	Pomen
CVE	Common Vulnerabilities and Exposures
UAF	Use-After-Free
glibc	GNU C library
tcache	Thread cache
ASLR	Address Space Layout Randomization
LIFO	Last In First Out
WebGL	Web Graphics Library

2 UVOD

Nevarnosti, izvirajoče iz napak pri upravljanju s pomnilnikom, ostajajo ključni izziv v programski opremi, razviti v programskih jezikih C in C++. Ta jezika omogočata neposreden dostop do pomnilnika in izjemno učinkovitost, hkrati pa nalagata odgovornost razvijalcem, da ročno skrbijo za dodeljevanje in sproščanje pomnilniških blokov.

Med najresnejšimi so ranljivosti, ki omogočajo okvaro pomnilnika in izvršitev poljubne kode; po-

sebej izstopa *dvojna sprostitev* (*double free*), ko isti pomnilniški blok neustrezno sprostimo več kot enkrat. Empirični podatki iz zbirke MegaVul (2006–2023), ki vsebuje 17,380 ranljivosti, kažejo, da je 59,27% vseh napak povezanih z upravljanjem pomnilnika, kar potrjuje trajno aktualnost problema in potrebo po sistematičnih pristopih k njegovi obravnavi [21].

Napake pri upravljanju s pomnilnikom imajo dolgo zgodovino v praksi in raziskavah. Rezultati preteklih del so kazali, da lahko prepis kazalcev ali napačno rokovanje z dodeljenimi bloki povzroči napake [1]. Kasnejše študije potrjujejo, da tovrstne ranljivosti niso zgolj teoretične, saj so orodja za odkrivanje prekoračitev kopice razkrila več deset doslej neznanih napak [4]. Takšne napake pogosto vodijo do nedefiniranega vedenja programa, ki se lahko izrazi v obliki sesutja, izgube podatkov ali celo vnosu zlonamerne kode v izvajalni tok.

Kljub različnim poskusom preprečevanja tovrstnih ranljivosti, primeri iz javno dostopne zbirke CVE [15] potrjujejo, da gre za še vedno aktualno in pogosto izrabljeno ranljivost. Napadi izkoriščajo tako stare kot tudi sodobne programske sisteme, kar kaže, da so obstoječi mehanizmi zaščite pogosto nezadostni ali neučinkoviti, še posebej ob kompleksnih scenarijih uporabe. Poleg tega so izkoriščanja teh napak zanimiva za napadalce, saj pogosto omogočajo eskalacijo privilegijev in obid varnostnih mehanizmov na ravni operacijskega sistema. V pričujočem članku v testnem okolju simuliramo in prikažemo izkoriščene ranljivosti dvojnega sproščanja in uporabe po sprostitvi pomnilnika ter demonstriramo metode za zaščito pred napadi tega tipa.

3 PREGLED PODROČJA

3.1 SORODNA DELA

Napadi, kot so *preliv kopice* (angl. heap overflow), *uporaba po sprostitvi* (angl. use-after-free) in *dvojna sprostitev* (angl. double free), so bili obsežno preučeni zaradi njihovega potenciala za okvara pomnilnika in izvajanje poljubne kode.

Temeljna raziskava na tem področju je delo Crispina idr., kjer so razvili metodologijo, imenovano *PointGuard*, za detekcijo zlonamernih programov, s fokusom na virusih. Njihova implementacija je bila efektivna za obrambo pred *prelivom kopice* [1]. Pogosto citiran znanstveni članek je tudi članek avtorja Kouwe in drugih. Osredotočajo se na detekcijo dostopov po sprostitvi s svojo rešitvijo DangSan. Rešitev je skalabilna ter podpira več nitne aplikacije [2].

Preliv kopice je ranljivost, kjer program zapiše več podatkov, kot jih je bilo predvidenih v prostoru na kopici. To lahko povzroči okvaro pomnilnika in ostale varnostne ranljivosti. Heelan idr. z implementacijo *Gollum* avtomatizirajo generacijo ranljivosti v tolmaču. V delu predstavijo moč implementacije z napadi v PHP in Python tolmačih [3]. V drugi smeri so raziskovalci iskali rešitve za detekcijo napadov med delovanjem programa. Pokažejo efektivnost delovanja z 17 resničnimi primeri v realnem svetu ter z najdbo 47 prej ne znanih ranljivosti [4]. Tematika je podrobneje obravnava v članku Gopala idr. [5] in delu He idr. [6]. Taki napadi so prisotni še danes, na primer CVE-2025-27091 [15] in CVE-2025-2531 [15].

Dvojne sprostitve se pojavijo, kadar je isti pomnilniški blok sproščen več kot enkrat, pogosto zaradi semantičnih napak programa. Take napake lahko povzročijo zrušitev programa, okvaro pomnilnika ali izvajanje poljubne kode. Maryam idr. predstavijo *fuzzing* metodo za detekcijo ranljivosti kopice v tej smeri, s čimer nadgradijo podobne programe iz prejšnjih raziskav. Rešitev dosežejo z kalkulacijo simboličnih poti in drugimi omejitvami za izvršljivo datoteko [7]. Na detekcijo tovrstnih napadov so se poglobili tudi Baradaran idr. Osredotočajo se na *detekcijo z enotno simbolično metodo* (angl. unit-based symbolic exection method). Program razdelijo na enote, ki lahko vsebujejo ranljivosti in so statično identificirane, glede na njihove specifikacije [8]. Tematika je podrobneje obravnava v članku Cabballera idr. [9] in članku Novarka idr. [10]. Primeri iz resničnega sveta, kot sta CVE-2025-2027 [15] in CVE-2025-32911 [15], kažejo, da so take ranljivosti prisotne tudi danes.

Dostopi po sprostitvi so tesno povezani z dvojnim sproščanjem in se pojavijo, ko program dostopa do pomnilnika po njegovi sprostitvi. Napadalec lahko nato prevzame nadzor nad tem pomnilniškim prostorom. UAF je vztrajno prisoten problem, zlasti v kompleksnih aplikacijah, kot so spletni brskalniki. Kailong idr. so se osredotočili na detekcijo takih napadov in so naredili prototip imenovan UAFDetector. Detekcijo dosežejo s sledenjem kazalcev ter *function summaries* pristopom [11]. Raziskovalci Quiang idr. so za obrambo pred takimi napadi raziskali rešitev Mpchecker, ki dinamično brani sistem z vmesnimi kazalci, ki jih imenujejo *Multi-Level Pointers*. Reši-

tev omogoča dostop do objektov samo z vmesnimi kazalci, ki jih avtomatsko sprosti ob sprostitvi objektov [12]. Tematika je podrobneje obravnava v članku Shena idr. [13] in delu Feista idr. [14]. Kot pri ostalih napadih, so tudi ti prisotni še danes, na primer CVE-2025-27730 [15]. Za širši pregled računalniških napadov bralcu priporočamo tudi širše preglede, npr. [16], [17].

Dirty COW (CVE-2016-5195) [15] je ranljivost, ki poveča privilegije v Linux jedru. Izkorišča *race condition*, ranljivost, ki se zgodi, ko več niti dostopa do istega podatka pri obravnavi sistemskega klica mmap. S tem lahko napadalec piše v *bralne* (angl. read-only) pomnilniške prostore, kar vodi do povišanja privilegijev. Napad deluje v Linux jedrih od verzije 2.x do 4.8.2.

3.2 DEFINICIJE

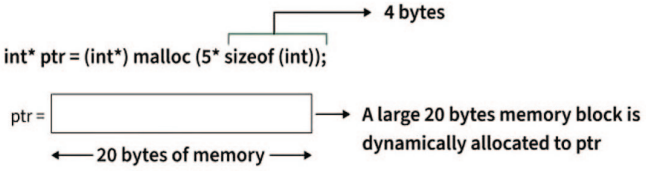
3.2.1 Kopica

Kopica (angl. heap) je prilagodljivo območje pomnilnika za shranjevanje večjih podatkovnih struktur in podatkov z dinamično življenjsko dobo. Od tradicionalnega upravljanja s pomnilnikom se razlikuje po tem, da z njo upravljamo eksplicitno sami, v jezikih kot sta Java in C# ponavadi preko operatorja new, v bolj nizkonivojskih jezikih kot je C, pa preko funkcije malloc(). Pozorni moramo biti na dejstvo da dodeljeni objekt/blok pomnilnika ostane v uporabi, dokler ga eksplicitno ne sprostimo. V nizkonivojskih programskih jezikih kot je C, je za to zadolžen programer z uporabo funkcije free(). Dodatna pozornost mora biti posvečena jezikom, kjer pomnilnik sproščamo sami, saj se hitro zgodi, da pomotoma naredimo preliv (ang. memory leak) ali kakšno drugo napako, ki podvrže naš program ranljivostmi, katero lahko napadalci izkoristijo.

3.2.2 Funkcija malloc()

Funkcija malloc() se uporablja, za dodeljevanje specifičnega števila bajtov pomnilnika. Funkcija pričakuje en argument, ki predstavlja število bajtov. Pozorni moramo biti na to, da malloc() samo rezervira prostor in ga ne inicializira, zato nimamo zagotovila, da bo dodeljeni blok prazen (angl. garbage values).

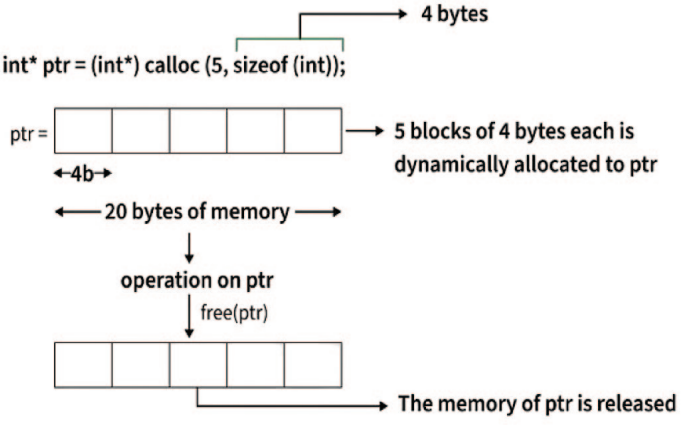
Funkcija malloc() vrne kazalec, ki kaže na dodeljeni blok pomnilnika, sledeči kazalec moramo, če ga želimo uporabiti, pretvoriti v ustrezni tip npr. int, char, itd. V primeru, da nimamo dovolj prostora na pomnilniku, bo funkcija malloc() vrnila prazen kazalec, zato je v praksi dobro preverjati, če je kazalec prazen (angl. NULL).



Slika 1: **Delovanje funkcije malloc [22].**

3.2.3 Funkcija free()

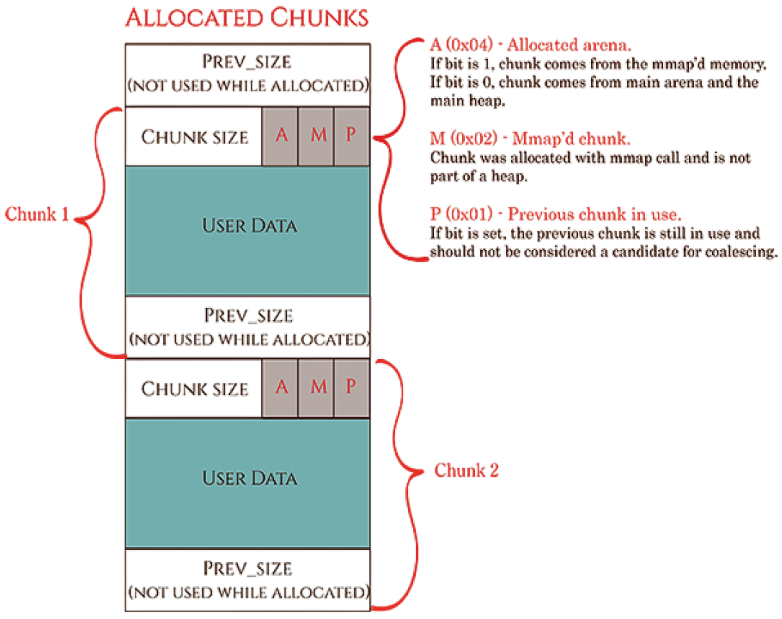
Funkcija free() se uporablja za sprostitev dodeljenih blokov. Njen edini argument je kazalec, ki kaže na lokacijo bloka v pomnilniku, katerega želimo sprostiti. Pri tem obdrži kazalec in samo sprosti blok na katerega je kazalec kazal, tako kazalec zdaj kaže na sproščeni blok. Pozorni moramo biti na dejstvo da funkcija free ne izbriše kazalca niti podatkov, ki so bili shranjeni na blok, ampak samo sporoči sistemu, da je zdaj ta blok prost (angl free) za druge dodelitve.



Slika 2: **Delovanje funkcije free [22].**

3.2.4 Arene

V večnitnih aplikacijah, mora kopični upravljalec (*angl. heap manager*) paziti, da več aplikacij ne dostopa do kopice hkrati. Rešitev so arene, ki delujejo kot različne kopice z svojimi strukturami. Arene so ločeni sklopi pomnilniškega prostora z lastnimi podatkovnimi strukturami za upravljanje dodeljevanje in sproščanje pomnilnika. Pri eno nitnih aplikacijah se uporablja samo glavna arena, ob dodajanju niti pa se ustvarijo sekundarne arene.



Slika 3: **Vizualna predstavitev bloka [23].**

3.2.5 Alokacija blokov

Podatke, ki dodelimo in sprostimo na kopico preko funkcij malloc() in free() se shranijo v enega izmed košev, kjer se rezervira prostor za zaglavje, naš podatek in velikost prejšnjega bloka. Zaglavje sestavljajo:

- trenutna velikost v Bajtih
- 3 zastavice A, M in P.
  - A (*Allocated arena*), je nastavljen na 0, če je blok v glavni areni in nastavljen na 1 sicer
  - M (*Mmap'd chunk*), je nastavljen na 1, če je bil blok alociran z klicem *mmap* in nastavljen na 0 sicer
  - P (*Previous chunk in use*) je nastavljen na 1, če je prejšnji blok še v uporabi, torej ni bil sproščen z funkcijo *free* in nastavljen na 0 sicer
- velikost prejšnjega bloka v Bajtih

Ko dodelimo podatke s funkcijo malloc() dobimo kazalec na prostor za te podatke, torej *user data*. Če hočemo delati z bloki direktno, torej hočemo, da kazalec kaže na zaglavje, lahko uporabimo makro *mem2chunk* ali za obratno makro *chunk2mem*.

3.2.6 Koši

Koši omogočajo učinkovito dodeljevanje in sproščanje pomnilnika. Bloki so vsebovani v koših. Vsaka arena vsebuje 5 tipov košev:

- **64 predpomnilniških košev *(angl. tcache bins)***
  Predpomnilniški koš je en povezan seznam največ 7 majhnih pomnilniških blokov, ki bloke ne združuje po sprostitvi.
- **10 hitrih košev *(angl. fast bins)***
  Hitri koš je namenjen pospeševanju časa alokacije za majhne pomnilniške bloke, tako da hrani predčasno sproščene bloke. Uporablja LIFO pristop, implementiran s povezanimi seznami, kar pomeni, da bo prostor zadnjega sproščenega bloka uporabljen pri novi alokaciji.
- **1 neurejen koš *(angl. unsorted bin)***
  Neurejen koš se uporablja kot medpomnilnik za kopičnega upravljalca, da pohitri dodeljevanje. Ko program sprosti pomnilniški blok, ga kopični upravljalec poskusi združiti s potencialno sproščenimi sosednjimi bloki, da naredi večji sprošče-

```
#define mem2chunk(mem) \\
  ((mchunkptr)tag_at (((char*)(mem) - CHUNK_HDR_SZ)))
#define chunk2mem(p) \\
  ((void*)((char*)(p) + CHUNK_HDR_SZ))
```

ni pomnilniški blok. Ta blok potem vstavi v neurejen koš. Ko program prosi za nov pomnilniški blok, upravljalec najprej pogleda v neurejen koš in kasneje v majhne in velike koše. Če prostora ne najde vse bloke iz neurejenega koša vstavi v primeren majhen ali velik koš.

- **62 majhnih košev** ***(angl. small bins)***
  Majhni koši so hitrejši od velikih, a počasnejši od hitrih. Vsak koš ima bloke enake dolžine: 16 B, 24 B, 32 B, ... Z maksimalno velikostjo 1024 B pri 64 bitnih sistemih, kar pomaga pri iskanju prostega prostora.
- **63 velikih košev** ***(angl. large bins)***
  Veliki koši, za razliko od majhnih, vsebujejo bloke, ki imajo skupno velikost v nekem razponu. Namesto enake velikosti, največji koš vsebuje bloke z velikostjo večjo od 1 MB.

4 METODOLOGIJA

4.1 Raziskovalno okolje

Raziskovalno okolje je sestavljeno iz enostavnega programa napisanega v programskem jeziku C, katerega namen je pisanje zapiskov, ki se shranjujejo na kopico. Program ima namenoma vgrajeno ranljivost kopice, s pomočjo katere lahko napadalec izvede napad dvojne sprostitve. To omogoča analizo in testiranje napada dvojne sprostitve v kontroliranem in znanem okolju.

V programu ločimo med dvema uporabnikoma: navadnim in administratorskim. Razlikujeta se le v tem, da administrator lahko spreminja administratorsko geslo, medtem ko ga navadni uporabnik ne more.

Uporabniki se po programu sprehajajo preko terminala, tako da napišejo pripadajočo številko pred podanimi možnostmi, katere lahko vidimo v sliki spodaj.

```
Welcome to the journal app!
[1] Make entry
[2] Delete entry
[3] Login as admin
[4] Print entries
[5] Exit
```

Slika 4: **Uporabniški vmesnik aplikacije.**

4.2 Struktura programa

Uporabniški vmesnik se vrti v neskončni zanki, katera se konča, ko uporabnik vnese v terminal številko pet, ki prekine delovanje programa. Uporabnik ima na voljo tudi možnost izdelovanja zapiskov, katere lahko tudi briše.

```
typedef struct Entry Entry;
typedef struct Entry {
    char* entry_content;
    Entry* next_entry;
    // ...
} Entry;

Entry* init_entry(char* entry_content,
                  time_t time) {
    // ...
}
void entry_delete(Entry* parent, Entry* entry,
        int entry_number, int current_number) {

    // ...
}
void entry_add(Entry* parent, char* entry_content,
               time_t time) {

    // ...
}
```

Izsek kode 1: **Struktura Entry**

Zapiski se shranjujejo v objekt, ki je poimenovan Entry (Izsek kode 1). Znotraj strukture najdemo 2 kazalca, eden kaže na vsebino zapiska, drugi pa na naslednji zapisek.

Kazalec je spremenljivka, ki kot vrednost shrani pomnilniški naslov dodeljene spremenljivke, v našem primeru bo kazal na uporabniški vnos, ki ga shranimo na kopico oziroma vsebino zapiska.

Funkcija Entry* init_entry(char* entry_content, time_t time) poskrbi za pravilno dodelitev prostora in inicializacijo spremenljivk ustvarjenega zapiska (Izsek kode 1).

Rekurzivni funkciji entry_delete(Entry* entry, int entry_number, int current_number) in entry_add(Entry* parent, char* entry_content, time_t time) se ukvarja z brisanjem zapiskov, oziroma sprostitvijo

```
void handle_action(AppState* app_state) {
    switch (app_state->last_action) {
        // ...
        case 3:
             if(app_state->is_admin == 0){
                printf("Hey admin, \
                    what is your password?\n");
            }
            // logout admin
            else {

                printf(GREEN "successfully \
                    logged out as admin :) \n"
                    COLOR_RESET);
                // ...
            }
            break;
        // ...

    }
}

void admin_login(AppState* app_state) {

  // ...
  if(strcmp(pass, app_state->user_input) == 0){
      printf(GREEN "successfully \
                  logged in as admin :) \n"
                  COLOR_RESET);
      app_state->is_admin = 1;
  }
  else {
      printf("wrong password, \
                  please try again \n");
      app_state->is_admin = 0;
  }
  handle_action(app_state);
}
```

Izsek kode 2: **funkciji za prijavo kot administrator**

dodeljenega prostora, ko uporabnik izbriše specifični zapisek (Izsek kode 1).

Uporabnik ima tudi možnost prijave kot administrator. Analizirajmo še kodo, ki preveri, če je uporabnik vpisal pravilno geslo (Izsek kode 2).

Funkcija handle_action(AppState* app_state) poskrbi za uporabniški vnos, katerega pridobimo preko terminala kot tabelo znakov tipa char (angl. array), na katero kaže kazalec user_input (Izsek kode 2).

Prijavo kot administrator ločimo v dveh primerih. Prva je ko uporabnik izbere tretjo možnost in je že prijavljen kot administrator. V tem primeru ga odjavimo iz sistema nazaj v navadnega uporabnika. Drugi primer je prijava, kot administrator. Takrat se pokliče funkcija admin_login(AppState* app_state), ki od njega zahteva administratorsko geslo (Izsek kode 2).

Najprej se bo uporabniku izpisalo ustrezno besedilo za vnos gesla:

Hey admin, whats your password?

Program potem čaka na vnos uporabnika in v primeru, da je vpisano geslo napačno izpiše sporočilo:

Wrong password, please try again!

Če je geslo pravilno, bo program uporabnika ustrezno prijavil v sistem kot administratorja.

Zaradi lažje predstave gesla namenoma ne šifriramo. Tukaj se opazi prva ranljivost našega programa. V realnem svetu, bi geslo, preden bi ga shranili v datoteko, ustrezno šifrirali.

4.3 Demonstracija napada

Preko programa opisanega v 4.1 bomo prikazali napad dvojne sprostitve. Napadalec lahko program napade z uporabo dvojne sprostitve. Najprej ustvari 2 zapiska, potem prvega sprosti, nato sprosti še drugega in nato ponovno prvega. Medtem mora sprostiti še en zapisek, da se izogne napaki zaporedne dvojne sprostitve, ki bi program izključila.

```
Welcome to the journal app!
[1] Make entry
[2] Delete entry
[3] Login as admin
[4] Print entries
[5] Exit
1
Entry:
aaa
--------------------
ENTRIES:
[1][freed: NO][Fri Apr 25 16:14:28 2025] aaa
--------------------
[1] Make entry
[2] Delete entry
[3] Login as admin
[4] Print entries
[5] Exit
1
Entry:
bbb
--------------------
ENTRIES:
[1][freed: NO][Fri Apr 25 16:14:28 2025] aaa
[2][freed: NO][Fri Apr 25 16:14:29 2025] bbb
--------------------
```

Slika 5: **Začetno ustvarjanje dveh zapiskov.**

```
[1] Make entry
[2] Delete entry
[3] Login as admin
[4] Print entries
[5] Exit
2
--------------------
ENTRIES:
[1][freed: NO][Fri Apr 25 16:14:28 2025] aaa
[2][freed: NO][Fri Apr 25 16:14:29 2025] bbb
--------------------
What entry to delete?
1
--------------------
ENTRIES:
[1][freed: YES]
[2][freed: NO][Fri Apr 25 16:14:29 2025] bbb
--------------------
```

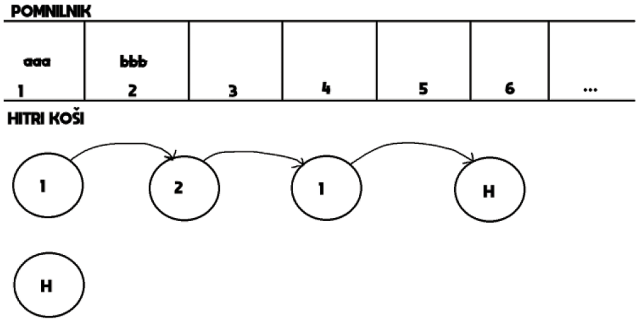
Slika 6: **Sproščanje prvega zapiska.**

```
[1] Make entry
[2] Delete entry
[3] Login as admin
[4] Print entries
[5] Exit
2
--------------------
ENTRIES:
[1][freed: YES]
[2][freed: NO][Fri Apr 25 16:14:29 2025] bbb
--------------------
What entry to delete?
2
--------------------
ENTRIES:
[1][freed: YES]
[2][freed: YES]
--------------------
```

Slika 7: **Sproščanje drugega zapiska.**

```
[1] Make entry
[2] Delete entry
[3] Login as admin
[4] Print entries
[5] Exit
2
--------------------
ENTRIES:
[1][freed: YES]
[2][freed: YES]
--------------------
What entry to delete?
1
--------------------
ENTRIES:
[1][freed: YES]
[2][freed: YES]
--------------------
```

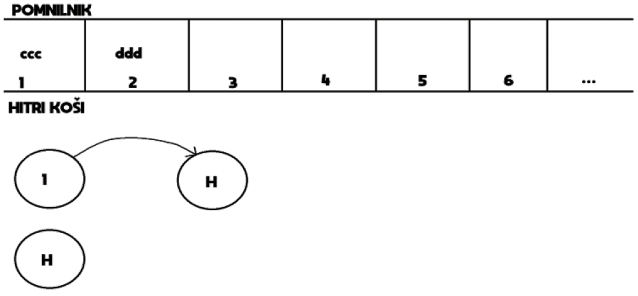
Slika 8: **Ponovno sproščanje prvega zapiska.**



Slika 9: **Stanje pomnilnika in hitrih košev po dvojni sprostitvi.**

```
[1] Make entry
[2] Delete entry
[3] Login as admin
[4] Print entries
[5] Exit
1
Entry:
ccc
--------------------
ENTRIES:
[1][freed: YES]
[2][freed: YES]
[3][freed: NO][Fri Apr 25 16:14:35 2025] ccc
--------------------
[1] Make entry
[2] Delete entry
[3] Login as admin
[4] Print entries
[5] Exit
1
Entry:
ddd
--------------------
ENTRIES:
[1][freed: YES]
[2][freed: YES]
[3][freed: NO][Fri Apr 25 16:14:35 2025] ccc
[4][freed: NO][Fri Apr 25 16:14:36 2025] ddd
--------------------
```

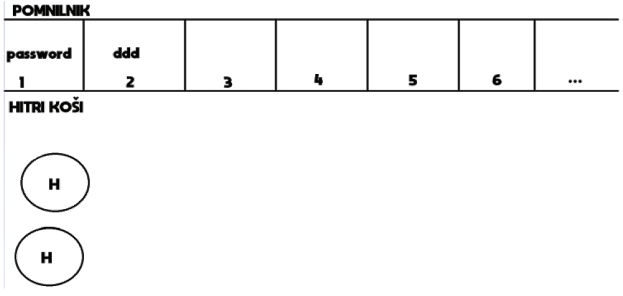
Slika 10: **Ustvarjanje še dveh zapiskov.**



Slika 11: **Stanje pomnilnika in hitrih košev po ustvaritvi še dveh zapiskov.**

```
[1] Make entry
[2] Delete entry
[3] Login as admin
[4] Print entries
[5] Exit
3
Hey adming ;), whats your password?
???
wrong passowrd, please try again
[1] Make entry
[2] Delete entry
[3] Login as admin
[4] Print entries
[5] Exit
4
--------------------
ENTRIES:
[1][freed: YES]
[2][freed: YES]
[3][freed: NO][Fri Apr 25 16:14:35 2025] password
[4][freed: NO][Fri Apr 25 16:14:36 2025] ddd
--------------------
[1] Make entry
[2] Delete entry
[3] Login as admin
[4] Print entries
[5] Exit
5
EXITING with exit status 0
```

Slika 12: **Neuspešna prijava in branje gesla.**



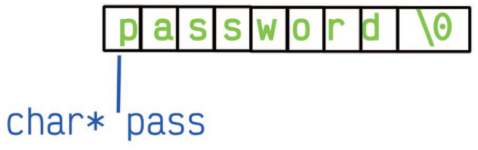
Slika 13: **Stanje pomnilnika in hitrih koševpo napadu.**

5 ANALIZA NAPADA

Po ponovnem sproščanju (Slika 8) vidimo, da sta na mestu 1 in 2 v pomnilniku dodeljeni vrednosti naših nizov “aaa” in “bbb”. Ob sprostitvi pomnilniške lokacije se dodeljeni nizi uvrstijo v isti koš, ker so podobne velikosti. Napadalec mora paziti na velikost njegovih zapiskov, saj se morajo ob sprostitvi zapiski uvrstiti v isti koš iz katerega kasneje dobi prostor geslo (Slika 10).

Ustvarimo še 2 zapiska (Slika 10). Ker ustvarjamo zapiske podobne velikosti, bomo dobili prostor od istega koša v katerega sta se sprostili pomnilniški lokaciji 1 in 2 (Slika 11). V hitrem košu je ostala še ena pomnilniška lokacija 1. Napadalec lahko to izkoristi tako, da se poskusi prijaviti kot administrator. V ozadju se pokliče funkcija admin_login (Izsek kode 3).

Funkcija admin_login() prebere geslo iz datoteke pass.txt in zanj rezervira ustrezno število bajtov. Prebrano geslo se shrani na kopico, tukaj moramo biti pozorni, da dodelimo en znak več kot je dolgo geslo, saj se v programskem jeziku kot je C tabele znakov končajo z “null terminatorjem”. Če za primer vzamemo besedo password, kljub temu da je beseda dolga 8 znakov, bi morali zanjo rezervirati 9 znakov, dodatni znak za “null terminator” (Izsek kode 3).



Slika 14: **Proces shranjevanja tabele znakov v programskem jeziku C**.

Funkcija dodeli prostor za geslo. Če je PASS_SIZE podobne velikosti, bo za dodelitev geslo dobilo prostor iz koša v katerem je pomnilniška lokacija 1. Posledično bo funkcija v ta pomnilniški prostor, nad katerim imamo nadzor, zapisala geslo. Vse kar mora napadalec narediti je, da se prijavi kot administrator in po neuspehu prebrati njegove zapiske. V primeru, da je PASS_SIZE primerne velikosti, bo funkcija

```
void admin_login(AppState* app_state) {
  char* pass = malloc(PASS_SIZE);
FILE* file = fopen("pass.txt", "r");
// ...
int size =
      fread(pass, sizeof(char), PASS_SIZE, file);
pass[size - 1] = '\0';

// ...
}
```

Izsek kode 3: **funkcija za prijavo kot administrator**

```
void entry_delete(Entry* parent, Entry* entry,
                  int entry_number,
                  int current_number) {

    if (entry_number == current_number) {
        // ...
        entry->is_freed = 1;
        free(entry->entry_content);
    }
    else {
    // ...
```

Izsek kode 4: funkcija za izbris zapiska

admin_login v ta prostor zapisala geslo, ki jo bo napadalec lahko prebral in se prijavil v aplikacijo kot administrator.

Napadalec to zlorabi in prebere geslo ter se prijavi kot administrator (Slika 12).

7 DISKUSIJA

Demonstracija napada dvojne sprostitve je razkrila številne posledice, ki jih napad ima kot so:

Kraja administratorskih privilegijev: Napadalec lahko prebere administratorjevo geslo in se prijavi kot administrator. Po prijavi lahko napadalec spremeni administratorjevo geslo in mu s tem prepreči dostop.

Nestabilnost sistema: Zaradi nepravilnega ravnanja s pomnilnikom (dvojna sprostitev) lahko pride do nepredvidenih vedenj aplikacije, vključno z zrušitvami, nedeterminističnim vedenjem in odkritjem dodatnih ranljivosti.

Širše posledice takih napadov: Takšni napadi ne vplivajo samo na delovanje aplikacije, ampak tudi na zaupanje uporabnikov, finančne izgube ter dolgoročni ugled.

Ugotovili smo, da so napadi na kopico zahtevni, saj napadalec potrebuje temeljito znanje in razumevanje njene strukture. V našem primeru je napadalec moral pazljivo dodeljevati prostor na pomnilniku preko ustvarjanja zapiskov, da so se shranili v enak koš, da je potem lahko pravilno prepisal shranjeno geslo.

Ni nujno, da je ranljivost prisotna znotraj našega programa, ampak lahko izvira iz zunanjih knjižnic ali drugih zunanjih virov, ki jih naš program potrebuje za delovanje. V praksi so te ranljivosti bolj pogoste, kot npr. ranljivost knjižnice WebGL za Chromium brskalnike [15].

Ostane nam še vprašanje kako se pred napadi zaščiti in kako jih rešiti? Vrnimo se v funkcijo entry_delete, ki skrbi za brisanje zapiskov (Izsek kode 4).

V funkciji opazimo uporabo spremenljivke *is_freed*, katero naš program uporablja za preverjanje, ali je bila vsebina zapiska sproščena ali ne. Gre za eno-

```
void entry_delete(Entry* parent, Entry* entry,
                  int entry_number,
                  int current_number) {
    //...
        if(entry->is_freed == 0){
            free(entry->entry_content);
            entry->is_freed = 1;
        }
        else {
            printf("DOUBLE FREE DETECTED ABORTING");
        }
    // ...
```

Izsek kode 5: posodobljena funkcija za izbris zapiska

stavno spremenljivko tipa int, katera ima lahko samo vrednosti 0, ki izraža vrednost false ali 1, ki izraža vrednost true (Izsek kode 4).

Rešitev je precej enostavna. Spremenljivko nastavimo na vrednost 1 šele potem, ko dejansko sprostimo vsebino s klicem funkcije free. S tem smo že na polovici rešitve. Kar nam še ostane je, da napišemo preprosti if stavek, ki preveri ali je zapisek že sproščen. Če to drži izpišemo opozorilo, da tega zapiska ni mogoče sprostiti, saj je že sproščen (Izsek kode 5).

Čeprav uvedba zastavice is_freed zmanjša možnost nenamerne dvojne sprostitve je ta pristop v praksi precej omejen. Zastavica ne preprečuje zlonamernega prepisovanja metapodatkov ali manipulacije z drugimi podatkovnimi strukturami na kopici. V večjih projektih je takšna rešitev pogosto nezanesljiva, saj se stanje objekta lahko spremeni na več mestih in zastavica ne zagotavlja dejanske zaščite pred logičnim napadom

Druga možna rešitev bi bila posodobitev verzije glibc, katera bi potem preko validacij uspešno zaznala dvojno sprostitev in tako prekinila izvajanje programa. Rešitev le preloži problem, ne pa nujno odpravi temeljne pomankljivosti, saj imajo tudi nove verzije glibc lahko varnostne ranljivosti.

Poglejmo si še bolj praktične rešitve, ki se uporabljajo v produkciji:

- **Uporaba pomnilniško varnih programskih jezikov:** Predvsem tisti, ki vključujejo samodejni sistem za upravljanje s pomnilnikom, na primer jeziki kot so Python, Java, C#, JavaScript in drugi visoko-nivojski jeziki. Za dodeljevanje pomnilnika poskrbi sam jezik in odgovornost ne leži več na programerju. Slabost tega je slabša hitrost in učinkovitost programa.
- **Uporaba alternativnih in varnejših implementacij funkcije malloc:** Kot alternative nam bolj varne implementacije omogočajo varnejši program, brez da bi zato žrtvovali hitrost in učinkovitost našega programa. Implementacije ponavadi funkcijo malloc popolnoma spremenijo kakor tudi delovanje dodeljevanja blokov.
- **Upoštevanje defenzivnega programiranja/dobrih praks:** Preko testiranja in igranja v peskovniku lahko odkrijemo potencialne napade in ranljivosti našega programa, kar omogoča zaznavanje napadov in hitro odzivanje programa na njih. Npr. blokiranje dostopa nepooblaščeni osebi. Tako poskrbimo, da kljub napadu, program še vedno deluje. To je predvsem pomembno v aplikacijah, kjer je pomembna celodnevna dostopnost, visoka varnost in hitrost.

Uporaba orodij za statično analizo kode: to nam omogoča odkrivanje ranljivosti že v razvojnem procesu, uporaba tako imenovanega fuzz testiranja, ki avtomatično generira nepredvidljive, naključne in neveljavne vnose za testiranje funkcionalnosti programa z namenom, da ga pokvari.

8 ZAKLJUČEK

V članku smo raziskali ranljivosti, povezane z napačnim upravljanjem s pomnilnikom ter predstavili praktično demonstracijo napada dvojne sprostitve. Poudarili smo, kako pomembno je razumevanje delovanja kopice, saj lahko napadalci s tem znanjem, izkoristijo tovrstne ranljivosti.

Analiza je pokazala, da lahko že preproste napake v kodi, npr. nepravilno ravnanje s kazalci, vodijo do varnostnih posledic. Prav tako je tudi opozorila, da je pomembna previdna uporaba zunanjih knjižnic.

Predstavili smo tudi potencialne rešitve, med glavnimi so preverjanje stanja kazalca pred sprostitvijo, uporaba novejših verzij knjižnic, uporaba orodij za statično analizo kode ter uporaba jezikov z samodejnim upravljanjem s pomnilnikom.

Predstavljeni ukrepi so koristni, vendar imajo omejitve. Preverjanje kazalcev pred sprostitvijo prepreči le enostavne napake, ne pa logičnih zlorab v kompleksnih sistemih. Posodabljanje knjižnic zmanjša tveganje, vendar ne izključi novih ranljivosti in prinaša težave z združljivostjo. Statična analiza pogosto daje lažne pozitivne rezultate ter ne zazna vseh napak med izvajanjem. Tudi jeziki z samodejnim upravljanjem pomnilnika odpravljajo le del težav, saj ranljivosti pogosto obstajajo v logiki aplikacije ali integraciji zunanjih modulov.

Obstajajo tudi kompleksnejši in nevarnejši napadi na kopico, ki jih nismo obravnavali. Primer takih so:

- **Tcache poisoning:** zloraba tcache koša za ponovno uporabo prostih blokov pomnilnika.
- **House of einherjar** in **House of force:** manipulacija z metapodatki kopice za prevzem nadzora nad dodeljevanjem pomnilnika.
- **Heap spraying:** množično polnjenje kopice s predvidljivimi podatki, kar poveča verjetnost uspeha napada.

Ti napadi so nevarnejši, saj zahtevajo poglobljeno znanje o notranjih mehanizmih upravljanja s pomnilnikom in so pogosto odporni na osnovne zaščitne ukrepe.

Smer razvoja se nagiba k rešitvam, ki temeljijo na ASLR tehnologiji, kar nakazujejo tudi novejše raziskave *Oreo: Protecting ASLR Against Microarchitectural Attacks (Extended Version)* [20].

Zaključimo lahko, da je razumevanje delovanje kopice in preprečevanje kopičnih ranljivosti ključno za zagotavljanje varnosti sistemov.

LITERATURA

[1] Cowan, C., Beattie, S., Johansen, J., & Wagle, P. (2003). PointGuardTM: Protecting pointers from buffer overflow vulnerabilities. In Proceedings of the 12th USENIX Security Symposium (pp. 91–104). USENIX Association.

[2] Erik van der Kouwe, Vinod Nigade, and Cristiano Giuffrida. 2017. DangSan: Scalable Use-after-free Detection. In Proceedings of the Twelfth European Conference on Computer Systems (EuroSys '17). Association for Computing Machinery, New York, NY, USA, 405–419. https://doi.org/10.1145/3064176.3064211

[3] Heelan, S., Melham, T., & Kroening, D. (2019). Gollum: Modular and greybox exploit generation for heap overflows in interpreters. In Proceedings of the ACM Conference on Computer and Communications Security (pp. 1689–1706). Association for Computing Machinery. https://doi.org/10.1145/3319535.3354224

[4] Jia, X., Zhang, C., Su, P., Yang, Y., Huang, H., & Feng, D. (2017). Towards efficient heap overflow discovery. In Proceedings of the 26th USENIX Security Symposium (pp. 989–1006). USENIX Association.

[5] Gopal, A. U. S., Soori, R., Ferdman, M., & Lee, D. (2023). TAILCHECK: A Lightweight Heap Overflow Detection Mechanism with Page Protection and Tagged Pointers. 17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23), 535–552. https://www.usenix.org/conference/osdi23/presentation/gopal

[6] L. He et al., "Automatically assessing crashes from heap overflows,"2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE), Urbana, IL, USA, 2017, pp. 274-279, doi: 10.1109/ASE.2017.8115640.

[7] Mouzarani, M., Sadeghiyan, B., & Zolfaghari, M. (2016). A smart fuzzing method for detecting heap-based vulnerabilities in executable codes. Security and Communication Networks, 9(18), 5098–5115. https://doi.org/10.1002/sec.1681

[8] Baradaran, S., Heidari, M., Kamali, A., & Mouzarani, M. (2023). A unit-based symbolic execution method for detecting memory corruption vulnerabilities in executable codes. International Journal of Information Security, 22(5), 1277–1290. https://doi.org/10.1007/s10207-023-00691-1

[9] Juan Caballero, Gustavo Grieco, Mark Marron, and Antonio Nappa. 2012. Undangle: early detection of dangling pointers in use-after-free and double-free vulnerabilities. In Proceedings of the 2012 International Symposium on Software Testing and Analysis (ISSTA 2012). Association for Computing Machinery, New York, NY, USA, 133–143. https://doi.org/10.1145/2338965.2336769

[10] Gene Novark and Emery D. Berger. 2010. DieHarder: securing the heap. In Proceedings of the 17th ACM conference on Computer and communications security (CCS '10). Association for Computing Machinery, New York, NY, USA, 573–584. https://doi.org/10.1145/1866307.1866371

[11] Zhu, K., Lu, Y., & Huang, H. (2020). Scalable static detection of use-after-free vulnerabilities in binary code. IEEE Access, 8, 78713–78725. https://doi.org/10.1109/ACCESS.2020.2990197

[12] Qiang, W., Li, W., Jin, H., & Surbiryala, J. (2019). Mpchecker: Use-After-Free Vulnerabilities Protection Based on Multi-Level Pointers. IEEE Access, 7, 45961–45977. https://doi.org/10.1109/ACCESS. 2019.2908022

[13] Zekun Shen and Brendan Dolan-Gavitt. 2020. HeapExpo: Pinpointing Promoted Pointers to Prevent Use-After-Free Vulnerabilities. In Proceedings of the 36th Annual Computer Security Applications Conference (ACSAC '20). Association for Computing Machinery, New York, NY, USA, 454–465. https://doi.org/10.1145/3427228.3427645

[14] Josselin Feist, Laurent Mounier, Sébastien Bardin, Robin David, and Marie-Laure Potet. 2016. Finding the needle in the heap: combining static analysis and dynamic symbolic execution to trigger use-after-free. In Proceedings of the 6th Workshop on Software Security, Protection, and Reverse Engineering (SSPREW '16). Association for Computing Machinery, New York, NY, USA, Article 2, 1–12. https://doi.org/10.1145/3015135.3015137

[15] cve.org "CVE". [Online]. Available: https://www.cve.org/

[16] Simon Hansman, Ray Hunt, A taxonomy of network and computer attacks, Computers & Security, Volume 24, Issue 1, 2005, Pages 31-43, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2004.06.011. (https://www.sciencedirect.com/science/article/pii/S0167404804001804)

[17] Mohan V. Pawar, J. Anuradha, Network Security and Types of Attacks in Network, Procedia Computer Science, Volume 48, 2015, Pages 503-506, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2015.04.126. (https://www.sciencedirect.com/science/article/pii/S1877050915006353)

[18] Liu, B., Olivier, P., & Ravindran, B. (2019). Slimguard: A secure and memory-efficient heap allocator. In Middleware 2019 - Proceedings of the 2019 20th International Middleware Conference (pp. 1–13). Association for Computing Machinery, Inc. https://doi.org/10.1145/3361525.3361532

[19] J. Ahn, K. Lee, C. Park, H. Moon and Y. Kwon, ŠwiftSweeper: Defeating Use-After-Free Bugs Using Memory Sweeper Without Stop-the-World,"in 2025 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2025, pp. 755-771, https://doi.ieeecomputersociety.org/10.1109/SP61157.2025.00131

[20] Song, S., Zhang, J., & Yan, M. (2024). Oreo: Protecting ASLR Against Microarchitectural Attacks (Extended Version). https://arxiv.org/abs/2412.07135

[21] Chao Ni, Liyu Shen, Xiaohu Yang, Yan Zhu, and Shaohua Wang. 2024. MegaVul: A C/C++ Vulnerability Dataset with Comprehensive Code Representations. In Proceedings of the 21st International Conference on Mining Software Repositories (MSR '24). Association for Computing Machinery, New York, NY, USA, 738–742. https://doi.org/10.1145/3643991.3644886

[22] Geeks for geeks. Dynamic Memory Allocation in C. URL: https://www.geeksforgeeks.org/c/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/

[23] Azeria labs. Arm Heap Exploitation. URL: https://azeria-labs.com/heap-exploitation-part-1-understanding-the-glibc-heap-implementation/

▪

Domen Breznik je študent 3. letnika 1. stopnje univerzitetnega študija na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Posebej ga zanimajo področja programske opreme in računalniške varnosti.

▪

Mark Novak je študent 2. letnika 1. stopnje univerzitetnega študija na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Posebej ga zanimajo področja razvoja videoiger, operacijski sistemi in računalniške varnosti.

▪

Matevž Pesek je izredni profesor in raziskovalec na Fakulteti za računalništvo in informatiko Univerze v Ljubljani, kjer je diplomiral (2012) in doktoriral (2018). Od leta 2009 je član Laboratorija za računalniško grafiko in multimedije. Od leta 2024 izvaja predmeta Varnost programov in Varnost sistemov, kjer se raziskovalno ukvarja s poučevanjem konceptov in organizacijo dogodkov s področja računalniške varnosti.

Iz Islovarja

Islovar je spletni terminološki slovar informatike, ki ga najdete na naslovu http://www.islovar.org. in ga ureja jezikovna sekcija Slovenskega društva Informatika. Vabimo vas k ogledu slovarja in prispevanju novih besed.

jezikovni model *(angl. language model, natural language model)* model za obdelavo besedila, ki na osnovi prejšnjih elementov besedila napoveduje naslednji element v besedilu

katedrálni razvójni modél -ega -ega -a m *(angl. Cathedral model)* način razvoja programja, pri katerem je delovna različica do izdaje dostopna le krogu razvijalcev

komponentni modél -ega -a m *(angl. component model* model, ki predstavlja elemente komponent in odvisnosti ter povezave med njimi

modél -a m *(angl. model)* poenostavljen in formaliziran opis sistema, pojava

modél entitét in povezáv -a -- -- -- m *(angl. entity-relationship model, ERM, entity relationship model)* model, ki opisuje entitete in povezave med njimi

mrežni podátkovni modél -ega -ega -a m *(angl. network data model)* podatkovni model, v katerem so podatki o entitetah prikazani kot navzkrižno povezane datoteke; prim. hierarhični podatkovni model, relacijski podatkovni model

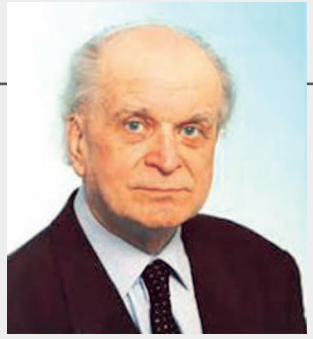
podátkovni modél -ega -a m *(angl. data model)*model, ki opisuje entitete in povezave med njimi v računalniški obdelavi podatkov

relacíjski podátkovni modél -ega -ega -a m *(angl. relational data model)* podatkovni model, ki je predstavljen z matematičnimi relacijami in dopušča operacije relacijske algebre; prim. hierarhični podatkovni model, mrežni podatkovni model

robustni model -ega -a m *(angl. robust model)* model, ki je neobčutljiv za manjše spremembe

žični model *(angl. wireframe, wireframe model, website wireframe, page schematic, screen blueprint)* prikaz razmestitve elementov na spletni strani, v spletni aplikaciji, ki služi oblikovalcem, urednikom in razvijalcem kot osnova pri postavljanju spletnega mesta oziroma aplikacije

In memoriam



prof. dr. **Anton Pavel Železnikar**
(1928–2026)

Uredništvo revije Uporabna informatika ter Slovensko društvo INFORMATIKA z globokim spoštovanjem in hvaležnostjo ohranjata spomin na prof. dr. Antona Pavla Železnikarja, enega ključnih pionirjev računalništva in informatike v Sloveniji in nekdanji Jugoslaviji, izjemnega znanstvenika, profesorja, vizionarja ter ustanovitelja revije Informatica.

Prof. dr. Železnikar je sodil med pionirsko generacijo, ki je postavila temelje sodobnega računalništva v Sloveniji. Prof. Železnikar se je po diplomi iz elektrotehnike na Univerzi v Ljubljani leta 1955 zaposlil na Institutu »Jožef Stefan«, kjer je deloval do leta 1980. V obdobju zgodnjega razvoja digitalnih tehnologij, ko je bilo raziskovanje še tesno povezano s strojno opremo, je svoje raziskovalno delo usmerjal v napredne koncepte, kot so algoritmi, formalni jeziki, prevajalniki in logika, kasneje pa tudi v paralelne računalniške sisteme in umetno inteligenco. Njegovo delo je pogosto presegalo časovni okvir svojega nastanka in nakazovalo smeri razvoja, ki so postale ključne šele desetletja kasneje.

Na Institutu »Jožef Stefan« je med letoma 1961 in 1978 vodil Odsek za digitalne tehnike ter med letoma 1968 in 1978 Odsek za elektroniko. S svojim delovanjem je pomembno prispeval k institucionalnemu razvoju in utrjevanju raziskovalnega področja računalništva v Sloveniji. Aktivno je sodeloval tudi v akademskem prostoru: na Univerzi v Ljubljani je bil leta 1982 imenovan za rednega profesorja, predaval pa je zlasti predmet iz področja prevajalnikov. Poleg tega pa je predaval tudi na Univerzi v Mariboru - filozofijo in teorijo informatike. S svojim pedagoškim in mentorskim delom je zaznamoval številne generacije študentov in raziskovalcev.

Leta 1980 je svojo karierno pot nadaljeval v podjetju Iskra Delta Computers, enem vodilnih visokotehnoloških podjetij v Sloveniji tistega časa. Tam je uspešno povezoval raziskovalno delo, razvojne procese in strateško načrtovanje. Med njegovimi pomembnimi prispevki izstopa koncept paralelnega procesiranja, ki je bil podlaga za razvoj inovativnega računalniškega sistema Triglav–Trident s tremi procesorji. Kasneje je deloval kot svetovalec generalnega direktorja ter kot član vodstva, odgovoren za strategijo raziskav in razvoja v okviru koncerna Iskra. Njegovo delovanje tako presega meje akademskega prostora in pomembno prispeva tudi k razvoju računalniške industrije.

Izjemno pomembna je bila tudi njegova mednarodna in strokovna dejavnost. Med letoma 1967 in 1975 je zastopal Jugoslavijo v mednarodni or-

ganizaciji International Federation for Information Processing (IFIP), leta 1971 pa je v Ljubljani organiziral svetovni kongres IFIP, ki je bistveno prispeval k mednarodni prepoznavnosti slovenske informatike. Leta 1976 je bil soustanovitelj Slovenskega društva INFORMATIKA in njegov prvi predsednik, pri čemer je aktivno sodeloval tudi v številnih drugih strokovnih in uredniških dejavnostih.

Njegov prispevek k razvoju znanstvene publicistike je posebej zaznamoval ustanovitev revije Informatica leta 1977, ki jo je dolga leta tudi urejal kot glavni urednik. Pod njegovim vodstvom se je revija uveljavila kot pomemben mednarodni znanstveni forum na področju računalništva, informatike, kibernetike in umetne inteligence. Njegova uredniška vizija in doslednost sta bistveno vplivali na oblikovanje kakovostnih standardov in dolgoročno prepoznavnost revije ter širše stroke.

Znanstveno delo prof. dr. Železnikarja obsega širok spekter raziskovalnih področij, med drugim teorijo preklopnih vezij, algebrsko logiko, algoritme, logiko, paralelne računalniške sisteme ter strategijo razvoja računalniške industrije, v kasnejšem obdobju pa tudi umetno inteligenco in raziskovanje zavesti. Objavil je več kot sto znanstvenih in strokovnih del v več jezikih, dve monografiji ter številna univerzitetna učna gradiva.

Za svoje delo je prejel številna priznanja ter sodeloval z uglednimi mednarodnimi akademijami in strokovnimi organizacijami. Njegova trajna zapuščina se kaže v institucijah, raziskovalnih usmeritvah in strokovnih skupnostih, ki jih je soustvarjal – na Institutu »Jožef Stefan«, v slovenskem raziskovalnem in industrijskem prostoru, v visokošolskem izobraževanju, v strokovnih združenjih ter v znanstvenem založništvu.

Prof. dr. Anton Pavel Železnikar ostaja zapisan kot ena osrednjih osebnosti slovenske informatike – kot znanstvenik izjemne širine, vizionar in soustvarjalec temeljnih razvojnih smeri stroke. Njegova dediščina ostaja trajno vtkana v razvoj področja in institucij, ki jih je pomagal oblikovati.

Spomin nanj ohranjamo z globokim spoštovanjem.

Slovensko društvo INFORMATIKA

Izpitni centri ECDL

ECDL (European Computer Driving License), ki ga v Sloveniji imenujemo evropsko računalniško spričevalo, je standardni program usposabljanja uporabnikov, ki da zaposlenim potrebno znanje za delo s standardnimi računalniškimi programi na informatiziranem delovnem mestu, delodajalcem pa pomeni dokazilo o usposobljenosti. V Evropi je za uvajanje, usposabljanje in nadzor izvajanja ECDL pooblaščena ustanova ECDL Fundation, v Sloveniji pa je kot član CEPIS (Council of European Professional Informatics) to pravico pridobilo Slovensko društvo INFORMATIKA. V državah Evropske unije so pri uvajanju ECDL močno angažirane srednje in visoke šole, aktivni pa so tudi različni vladni resorji. Posebno pomembno je, da velja spričevalo v 148 državah, ki so vključene v program ECDL. Doslej je bilo v svetu v program certificiranja ECDL vključenih že preko 16 milijonov oseb, ki so uspešno opravile preko 80 milijonov izpitov in pridobile ustrezne certificate. V Sloveniji je bilo doslej v program certificiranja ECDL vključenih več kot 18.000 oseb in opravljenih več kot 92.000 izpitov. V Sloveniji sta akreditirana dva izpitna centra ECDL, ki imata izpostave po vsej državi.

AGORA

Micro Team

Znanstveni prispevki

Strokovni prispevki

Informacije

ISSN 1318-1882