SVEUČILIŠTE U ZAGREBU
FILOZOFSKI FAKULTET
ODSJEK ZA INFORMACIJSKE I KOMUNIKACIJSKE ZNANOSTI
2017./2018.

Marko Požega

Sigurnost i penetracijsko testiranje mrežnih aplikacija

(studija slučaja)

Diplomski rad

Mentor: dr. sc. Kristina Kocijan, doc.

Zagreb 2018.

Zahvala

Zahvaljujem svojoj mentorici dr. sc. Kristini Kocijan, doc. što me je nesebično motivirala, podupirala i mentorirala tijekom mog cjelokupnog akademskog obrazovanja. U trenutcima kada je bilo teško, lijepo je znati da sam pored profesora imao i prijatelja. Hvala!

Zahvaljujem se anonimnoj ustanovi i njihovoj informatičkoj službi (na čelu s voditeljem službe) na ustupanju prava za izvođenje istraživanja u ovome radu.

Zahvaljujem se svim prijateljicama i prijateljima, kolegicama i kolegama koji su mi pomogli da se na ovom putu izgradim kao osoba i kao akademski građanin.

Hvala curi Ivi, koja mi je bila potpora i motivacija u najtežim trenutcima.

Zahvaljujem se svojim roditeljima na ukazanom strpljenju, povjerenju, inspiraciji i pomoći tokom svih ovih godina. Ovaj rad posvećen je vama.

Sadržaj

Sažetak

U ovome radu bavio sam se problematikom i važnosti informacijske sigurnosti u modernim mrežnim aplikacijama kroz realistični primjer penetracijskog testiranja. Ovo područje računarske i informacijske znanosti od velikog je značaja privatnim i državnim organizacijama koje grade integritet, kredibilitet i konkurentnost na tržištu usmjerujući svoje poslovanje i znanje u mrežne aplikacije. Cilj rada je postepeno upoznati čitatelja sa svim apstraktnim slojevima na kojima se temelje mrežne aplikacije, kako bi u konačnici došli do definiranja sigurnosnih ranjivosti te njihovih otkrivanja i saniranja. U svrhu boljeg razumijevanja, u radu su detaljno obrađene teme računalnih mreža, Interneta, kriptografije, kategorizacije sigurnosnih propusta te faza, konvencija i standarda penetracijskog testiranja. Predstavljeni su i rezultati anonimnog testiranja mrežne aplikacije koristeći nekomercijalni alat za penetracijsko testiranje *Metasploit*. Ovaj rad može biti od koristi kako potencijalnim penetracijskim testerima, (etičkim hakerima) tako i razvojnim programskim inženjerima mrežnih aplikacija.

Ključne riječi: *mrežne aplikacije, računalne mreže, Internet, informacijska sigurnost, etičko hakiranje, penetracijsko testiranje*

Security and penetration testing of web applications (case study)

Abstract

This paper deals with the issue and importance of information security in modern web applications through a realistic example of penetration testing. This area of computer and information science is of great importance to private and state organizations that build integrity, credibility and market competitiveness by directing their business and knowledge into web applications. The aim of the paper is to gradually introduce readers to all abstract layers on which network applications are based in order to finally define security vulnerabilities and their detection and remediation. After selecting topics such as computer networks, the Internet, cryptography, security vulnerabilities categorization, phases, conventions, and penetration testing standards, I've conducted a penetration test of the anonymous network application using the non-commercial Metasploit testing framework. This work can be of use to both potential penetration testers (ethical hackers) and web developers.

Key words: *web application, computer newtwork, Internet, information security, ethical hacking, penetration testing*

1. Uvod

Rapidni razvoj i rast informacijske tehnologije, informacijskih mreža, informatizacija te digitalizacija gradiva kroz nekoliko proteklih desetljeća rezultirali su transferom i perzistiranjem poslovne logike (znanja općenito) na dobro nam poznatoj i najraširenijoj mreži - Internetu, najčešće u obliku mrežne aplikacije (*engl. Web application*). Budući da svako znanje nije javno (Tuđman et al., 1991), potrebno je osigurati i regulirati privatnost znanja odnosno informacija koje egzistiraju na nekoj mrežnoj aplikaciji. Ulaganje u održavanje i razvoj mjera informacijske sigurnosti privatnim i državnim organizacijama daje kredibilitet i konkurentnost na tržištu. Kompleksnost mrežnih sustava iziskuje stručnost i specijalizirano znanje potrebno za njihov razvoj i održavanje. Posebna skupina stručnjaka zadužena je za otkrivanje nepravilnosti u radu i sigurnosnih propusta sustava, a nazivamo ih hakerima (*engl. Hacker*). Ovisno o njihovim etičkim ciljevima i moralnoj orijentaciji, hakerske radnje mogu bili dobrobitne ili maliciozne, s toga se u ovom području koristi analogija bijelih (*engl. White hat*) i crnih šešira (*engl. Black hat*) (Baloch, 2015).

U ovome radu na glavu ću staviti bijeli šešir te opisati moguće prijetnje modernim mrežnim aplikacijama gledano iz perspektive korisnika aplikacija, razvojnog inženjera te u konačnici penetracijskog testera. Kroz poglavlja ćemo prelaziti iz nekih viših slojeva apstrakcije u detaljne tehničke opise procesa, od generičkog koncepta računalnih mreža do detaljnih definicija i provedbe sigurnosnih napada.

U drugom poglavlju rada govorit ću o računalnim mrežama općenito s većim naglaskom na računalnu mrežu *Internet* koji čini infrastrukturalni i softverski temelj moderne mrežne aplikacije. Detaljnije ću pojasniti suvremene protokole koji djeluju na aplikacijskom sloju računalne mreže, posebno u kontekstu ostvarivanja sigurne šifrirane veze gdje ću se dotaknuti nekih elementarnih koncepata moderne kriptografije.

Treće poglavlje namijenio sam konciznom definiranju pojmova i najfrekventnijih sigurnosnih ranjivosti mrežnih aplikacija. Svakoj od ranjivosti spomenutih u ovome radu pristupit ću s gledišta korisnika aplikacije, kao i s gledišta razvojnog inženjera mrežnih

aplikacija, skrećući pažnju na poželjne prakse defanzivnog programiranja i metode suzbijanja te neutraliziranja sigurnosnih prijetnji.

Četvrto i peto poglavlje čine simbiozu teorije penetracijskog testiranja sa studijom slučaja i prezentiranjem rezultata penetracijskog testiranja anonimne mrežne aplikacije. U tom dijelu rada korak po korak proći ćemo kroz sve faze penetracijskog testiranja na generičkoj konceptualnoj razini, te na nižoj tehničkoj razini opisom i uporabom konkretnih alata za penetracijsko testiranje.

Kao ishod ovoga rada, očekujem da će čitatelj steći koncizan pregled problematike ovoga područja, dok ću s druge strane, studijom slučaja anonimne mrežne aplikacije, pružiti besplatnu uslugu penetracijskog testiranja s kojom će se nadamo spriječiti neki potencijalni sigurnosti propusti.

2. Arhitektura mrežnih aplikacija

U ovome poglavlju razložit ću i objasniti esencijalne komponente potrebne za postojanje jedne mrežne aplikacije, njezin rad te sam proces odnosno apstraktni mehanizam interakcije korisnika s aplikacijom. Iz pojmova poput računalnih mreža, mrežnih protokola, baza podataka, kriptografskih algoritama i ostalih pokušat ću izolirati nužne informacije potrebne za razumijevanje konteksta ovoga rada. Nekim čitateljima se sadržaj ovog poglavlja može činiti rudimentarnim i dobro poznatim, no ipak smatram nužnim navesti i pojasniti svu terminologiju vezanu uz tematiku koju primarno obrađujem.

2.1. Računalne mreže

Kako bi osoba mogla razmijeniti informaciju, znanje ili iskustvo s drugom osobom, nužno je obostrano ili bar jednostrano poznavanje prirodnog govornog, ili bilo kojeg drugog sporazumnog jezika, kulture ili nekog skupa definiranog ponašanja druge osobe. Istu apstrakciju koriste i naivna računala koja između sebe moraju prenositi nizove njima (ne)prirodnog strojnog jezika. S dizajniranjem takvih kompleksnih tehnoloških sustava koji omogućavaju prijenos informacijskih sadržaja između dvaju ili više samostalnih računala bavi se posebna grana računarstva, a opisani sustav naziva se *računalna mreža* (*engl. Computer network*) (Radovan, 2010). Trenutno najveća svjetska mreža je *Internet,* koju konzumiraju današnje mrežne aplikacije i koja će biti središnja tema ovog poglavlja.

U nastavku poglavlja objasnit ćemo osnovni rad i kategorizaciju računalnih mreža. Također ćemo objasniti nama važnu klijent-poslužitelj komunikaciju te se pozabaviti konceptom Interneta, modernih šifriranih i nešifriranih aplikacijskih protokola, kao i slojevima modernih računalnih mreža s naglaskom na slojeve više hijerarhijske razine, na kojima ćemo u daljnjim poglavljima vršiti penetracijsko testiranje.

2.1.1. Generički koncept i terminologija

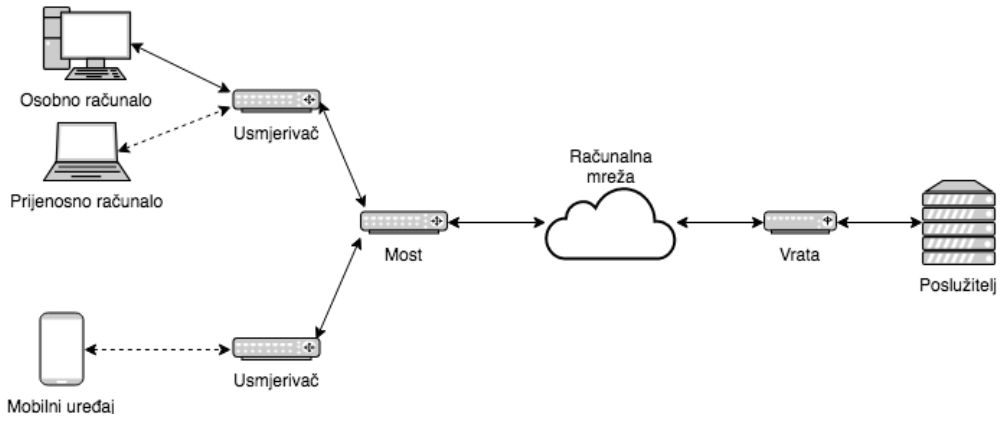
Započnimo s nekim temeljnim pojmovima poput klijent (*engl. Client*) i poslužitelj ili opslužitelj (*engl. Server*). Spomenute entitete obično tvore softverski sustavi (programi) koji su smješteni na nekom fizičkom uređaju (računalu). Klijent šalje neki zahtjev poslužitelju koji isti zahtjev izvršava te potom klijentu, ovisno o prirodi zahtjeva, šalje ili ne šalje neki povratni rezultat. To je ujedno najraširenije načelo računalne komunikacije povrh kojeg su temeljene tradicionalne mrežne aplikacije. Neki od primjera takve komunikacije su dohvaćanje mrežnih stranica, razmjena multimedijalnog sadržaja između dvaju računala, preuzimanje i izravno prikazivanje video sadržaja (*engl. Streaming*) i slično. Budući da smo u uvodu poglavlja računalnu mrežu definirali kao mrežu od minimalno dvaju povezanih računala, onda je naša klijent-poslužitelj komunikacija - računalna mreža.

S obzirom na to da je računalna mreža zapravo analogija paukove mreže koja ima svoja sjecišta i veze, tako i računalne mreže imaju svoje čvorove i veze koje ostvaraju prijenos podataka među njima. Čvorovima definiramo sva računala koja su uključena u mrežu. Računala na kojima rade poslužitelji i na koja se izravno vezuju klijenti (korisnici), nazivamo domaćinima (*engl. Hosts*). Primjećujemo da je pri definiciji računalne mreže sama mreža uvjetovana s dvjema ili više međusobno povezanih računala. Što znači da naša veza klijent-poslužitelj ne mora nužno imati samo dva spomenuta entiteta, već u komunikaciji može sudjelovati niz (mreža) međusobno vezanih čvorova. Takav tip mreže nazivamo mrežom s neizravnom vezom (*engl. End to end connection*), u protivnom vezu nazivamo izravnom (*engl. Point to point connection*). Čvorovi koji ima namjenu ostvarivanja prijenosa sadržaja (podataka) u mreži nazivamo prijenosnicima. Termin *prijenosnik* zapravo objedinjuje nekoliko elektroničkih naprava koje izvode usmjeravanje i prijenos jedinica podataka, kao što su preklopnici (*engl. Switch*), mostovi (*engl. Bridge*), koncentratori (*engl. Hub*), vrata (*engl. Gateway*), i usmjerivači (*engl. Router*). Za potrebe ovoga rada nije nam potrebno detaljno poznavanje spomenutih naprava i njihovih funkcionalnosti tako da ću se prema njima u nastavku rada referirati spomenutim objedinjenim nazivom. Budući da je svaki čvor računalne mreže fizičko računalo, ono mora biti jedinstveno označeno kako komunikacija ne bi bila ostvarena s pogrešnim

računalom. Kako bi to postigli, svako računalo posjeduje jedinstvenu fizičku MAC adresu (*engl. Media Access Control address*) unikatnu u cijelome svijetu te jedno ili više virtualnih odnosno logičkih adresa (adresa protokola) koje su jedinstvene u nekoj računanoj mreži. Primjer jedne fizičke MAC adrese koja je strukturirana iz šest parova heksadecimalnih znakova (ukupno 48 bitova) je *e5:0:37:d:d0:9c*. Kako bi jedno fizičko računalo moglo biti član više računalnih mreža, razvijeni su sustavi za adresiranje na razini računalnih mreža koji vrše pretvorbe logičkih adresa u fizičke i obratno. Logičke adrese mijenjaju se dinamički svakim pristupanjem i odstupanjem iz računalne mreže, pa se tako i tablice sustava za adresiranje ažuriraju sukladno promjenama strukture mreže.

Nakon što smo definirali moguće entitete i uloge koje mogu preuzeti čvorovi računalne mreže, spustit ćemo se na hardversku (fizičku) razinu i definirati što su, te od čega se sastoje veze koje povezuju čvorove. Veze između čvorova ostvaruju se pomoću raznih nosioca (medija) signala. Neki od najkorištenijih medija su opletena parnica (*engl. Twisted pair*) kakvu obično nalazimo u fiksnim telefonskim sustavima, koaksijalni kabel (*engl. Coaxial cable*), optički kabeli te razni uređaji namijenjeni bežičnom prijenosu signala putem radiovalova, mikrovalova, infracrvenih zraka i slično. Unikatna zadaća svih ovih medija svodi se na što precizniji i brži prijenos elektromagnetskih ili svjetlosnih signala (nizova bitova) na što veće udaljenosti. Ovaj sloj apstrakcije u računalnoj mreži ujedno nazivamo i fizički sloj o kojem ću nešto reći u narednom poglavlju. O načinu kodiranja, otkrivanju i ispravljanju pogrešaka prilikom prijenosa signala na fizičkom sloju brinu se jedni od viših slojeva računalne mreže.

Vrlo je važno naglasiti da računalne mreže rekurzivno mogu inkapsulirati jednu ili više manjih računalnih mreža. Računalna mreža koja se sastoji od više računalnih mreža univerzalno se naziva engleskim terminom *internetwork* ili skraćeno *internet*, a najveći i najstariji oblik takve računalne mreže naziva se Internet. Obratimo pažnju na to što nam govori veliko, a što malo početno slovo ove riječi. Vizualni prikaz opisanog osnovnog generičkog koncepta i elemenata računalne mreže možemo vidjeti na slici 1. Budući da u stručnoj literaturi računalne mreže obično bivaju ilustrirane simbolom oblaka, oblak na slici 1 označava inkapsuliranu računalnu mrežu sačinjenu od različitih čvorova i veza (oblik strelice).



Slika 1. Ilustrativni prikaz generičke računalne mreže i odnosa klijent-poslužitelj

2.1.2. Klasifikacija

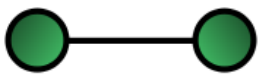
Ne postoji univerzalna podjela računalnih mreža, već ih možemo kategorizirati prema pojedinim osobinama i kriterijima. Podjela računalnih mreža obično se odvija prema modelu komunikacije koju koristi, topologiji te njenoj (geografskoj) opsežnosti.

U prethodnom poglavlju započeli smo priču o jednom od sustava komunikacije unutar računalnih mreža, a to je **model klijent-poslužitelj**. U ovome sustavu jedan krajnji čvor uvijek ima ulogu klijenta dok drugi krajnji čvor (domaćin) ima ulogu poslužitelja. Poslužitelji su obično veća računala blago različite arhitekture od osobnih računala koja su stalno aktivna i vezana na mrežu. Na domaćine (računala) poslužitelja obično je instaliran neki oblik softvera koji razumije protokol i načela modela komunikacije unutar računalne mreže kojeg možemo djelomično prevesti kao *mrežni poslužitelj* (*engl. Web server*). U tradicionalnim mrežnim aplikacijama aplikacijski sloj komunicira s aplikacijskim slojem (mrežnim serverom) poslužitelja. Detaljnije o ovom specifičnom tipu komunikacije putem HTTP protokola govorit ću u četvrtom potpoglavlju. Drugi model komunikacije unutar računalnih mreža naziva se **P2P** (*engl. Peer to peer*) gdje pojam *peer* definira entitet koji je iste vrste i hijerarhijske razine kao i drugi entiteti. U ovom sustavu svaki od čvorova mreže, kojih može biti neograničeni broj, čine računalnu mrežu u kojoj su svi čvorovi ravnopravni te u isto vrijeme mogu poprimiti ulogu klijenta i poslužitelja. Iz prethodne definicije možemo zaključiti da je ovakav sustav decentraliziran i iziskuje posebnu mrežnu

(logičku) topologiju mrežastog oblika o kojoj ću govoriti u nastavku teksta. Ovaj model komunikacije je znatno manje zastupljen među mrežnim aplikacijama, iako u posljednje vrijeme sve više nalazi svoju primjenu u praksi. Neki od praktičnih rješenja koje nudi ovaj model su decentralizirano distribuiranje datoteka, internetska telefonija te prijenos televizijskog programa pomoću Interneta. Iako se naizgled ovaj model komunikacije čini jako korisnim i posjeduje zanimljive osobine, također posjeduje i određene slabosti koje su zapravo najviše uvjetovane mrežnom topologijom koju koristi (više o tome u nastavku teksta). U novije vrijeme neke aplikacije također koriste **kombinirane elemente klijent-poslužitelj modela i P2P modela** čime se definira nova miješana (hibridna) struktura. Primjer takve aplikacije mogao bi izgledati poput procesa gdje je u jednoj fazi izvršenja mrežne aplikacije potrebno od nekog centraliziranog poslužitelja zatražiti adresu računala s kojim u drugoj fazi želimo razmjenjivati sadržaj putem P2P modela.

Drugi oblik podjele računalnih mreža je prema njenoj fizičkoj topologiji odnosno rasporedu i vezama između čvorova. Prema Bicsi (2002) u teoriji i praksi poznajemo osam bazičnih fizičkih topologija, a to su (u slobodnom prijevodu): „od točke do točke“ (*engl. Point to point*), magistrala ili sabirnica (*engl. Bus*), zvjezdasta (*engl. Star*), prstenasta ili cirkularna (*engl. Ring*), stablasta (*engl. Tree*), mrežasta (*engl. Mesh*), hibridna (*engl. Hybrid*) i lanac (*engl. Daisy chain*).

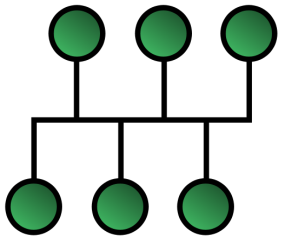
Topologija „**od točke do točke**“ najjednostavnija je mrežna topologija i predstavlja koncept izravnog povezivanja dvaju čvorova putem nekog fizičkog medija (slika 2). Ovakav tip mreže može se dinamički stvoriti i uništiti kada komunikacija između dva kraja više nije potrebna, stoga možemo reći da je ovaj oblik mreže često virtualan. Takav proces virtualizacije komunikacijskog kanala između dvaju krajnjih korisnika najčešće nalazimo u modernoj telefoniji.



Slika 2. Struktura topologije „od točke do točke“

Magistrala ili ***sabirnica*** oblik je mrežne topologije gdje se koristi jedan vodič (sabirnica) na kojeg se spajaju svi čvorovi te mreže (slika 3). Takva mreža se vrlo lako

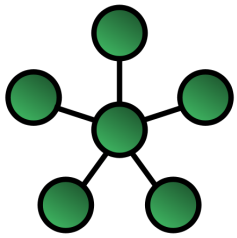
proširuje na način da se računalo može priključiti na sabirnicu te time postaje čvorom mreže. Prednost ovog oblika mreže je vrlo jeftina implementacija i mogućnost širenja mreže jer se zasniva na jednom vodiču. S druge strane, zbog svoje centraliziranosti, rad mreže može biti obustavljen oštećenjem rada sabirnice jer je to jedini komunikacijski kanal između svih čvorova u mreži. Također jedan od glavnih nedostataka ovog mrežnog oblika je nemogućnost multipleksiranja signala čvorova mreže. Drugim riječima, jedan čvor može komunicirati s drugim čvorom isključivo onda kada je sabirnica slobodna i ne postoje drugi signali koji se prenose sabirnicom. Ovo ograničenje postoji zbog lakog nastajanja kolizije signala i teškog odnosno gotovo nemogućeg detektiranja autentičnosti signala. Ovaj oblik mreže popularan je u lokalnim mrežama i koristi ga najuspješnija lokalna mreža *Ethernet.* U Ethernetu je sabirnica obično koaksijalni kabel, ali mogu se koristiti i drugi vodiči. Sabirnica smije biti ukupne dužine do 2500 metara te smije sadržavati najviše 1024 čvora.



Slika 3. Struktura sabirnice[1]

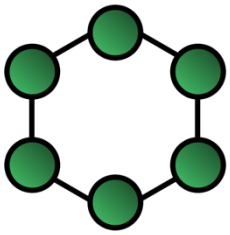
Zvjezdasta struktura je tip strukture koja se sastoji od konačnog broja čvorova povezanih na centralizirani čvor koji ima ulogu prijenosnika (slika 4). Prijenosnik je najčešće implementiran kroz uređaj preklopnika, ali to u nekim slučajevima može biti i koncentrator. Integritet mreže ostaje netaknut ako jedan od krajnjih čvorova mreže prestane s radom, međutim ako se to dogodi s prijenosnikom odnosno centralnim čvorom, mreža gubi svoju funkciju. Ova mrežna struktura također pogoduje izgradnji lokalnih mreža.

[1] Slika preuzeta sa: https://kids.kiddle.co/Network_topology



Slika 4. Zvjezdasta struktura[2]

Prstenasta struktura je sama po sebi vrlo deskriptivna i, kao što možemo zaključiti iz samog imena, prati decentralizirani uzorak koji tvori oblik prstena ili jednostavnije rečeno zatvorenog kruga (slika 5). Svaki čvor ove strukture povezan je s dva susjedna čvora jednostrukom ili dvostrukom (sekundarnom) vezom koja služi kao rezervna poveznica u slučaja kvara primarne fizičke veze. Ovisno o smjeru cirkuliranja podataka, ovakav tip mreže može sadržavati jednosmjerne ili dvosmjerne veze. Podaci koji se prenose prstenom prolaze kroz svaki čvor mreže što stvara glavni nedostatak ovog tipa jer ako jedan od tih čvorova nije u mogućnosti prenositi podatke, mreža prestaje biti funkcionalna. Također, onaj čvor koji posjeduje najmanju propusnost podataka, postaje usko grlo sustava. S druge strane, ova mreža je popularna pri kreiranju lokalnih mreža gušćeg prometa jer nudi bolje performanse od strukture sabirnice. Tri najpoznatije mreže prstenastog tipa su IBM Token Ring, FDDI (*engl. Fiber Distributed Dana Interface*) te RPR (*engl. Resilient Packet Ring*).

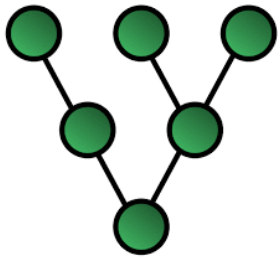


Slika 5. Prstenasta struktura[3]

Stablasta topologija manifestira se kroz formu stabla (slika 6). Stablo sadrži jedan centralni čvor koji je najviši u hijerarhijskom poretku čvorova i na njega se rekurzivno

[2] Slika preuzeta sa: https://kids.kiddle.co/Network_topology
[3] Slika preuzeta sa: https://kids.kiddle.co/Network_topology

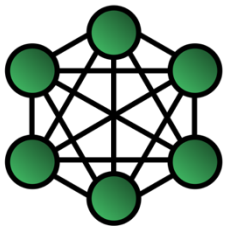
nastavljaju vezati čvorovi tvoreći nove hijerarhijske razine. Da bi neka mreža imala odliku stablaste toplogije, mora sadržavati minimalno tri hijerarhijska sloja. Glavna prednost ovakve strukture je ta što je mrežu vrlo lako proširiti, te je samim tim izolacija čvorova s pogreškom u prijenosu vrlo laka. Nedostatak je naravno što jedan neispravan čvor ili veza skraćuju obujam mreže za se podređene grane tog čvora ili veze.



Slika 6. Stablasta struktura[4]

Mrežasta topologija definira strukturu mreže u kojoj svaki čvor može imati izravne mreže prema jednom, više ili svim čvorovima mreže (slika 7). U instanci mreže u kojoj su svi čvorovi međusobno povezani izravnim vezama nazivamo potpunom mrežom (*engl. Full mesh*) dok krnji slučaj nazivamo parcijalnom mrežom (*engl. Partial mesh*). Prednost ove topologije je izravna povezanost sa svim čvorovima mreže čime se izbjegava korištenje nekih od prijenosnika, te ostvaruje visoka propusnost u komunikacijskom kanalu. Glavni nedostatak mreže je očita zalihost veza između čvorova što rezultira porastom cijene implementacije jer broj veza u odnosu na svaki novi čvor raste kvadratno. Potpuno mrežasta struktura koristi se rijetko zbog prethodno navedenih činjenica, tako da je njezina implementacija realizirana tamo gdje je krajnje nužna međusobna povezanost i integritet mreže, te naravno gdje broj čvorova ne prelazi neku razumnu (isplativu) brojku. Mnogo češći primjer implementacije ove topologije je kroz parcijalno povezane čvorove kojima se pokušava optimizirati rad mreže.

[4] Slika preuzeta sa: https://kids.kiddle.co/Network_topology



Slika 7. Potpuna mrežasta struktura[5]

Hibridne topologije nastaju u trenutku kada se dvije mreže različitih fizičkih topologija međusobno povežu. U praksi može postojati velik niz različitih kombinacija međusobno povezanih mreža pa se čak za one najčešće uvodi posebna nomenklatura.

***Daisy chain* topologija** zapravo predstavlja koncept koji kaže da bi svaki čvor mreže trebao moći pristupiti bilo kojem drugom čvoru u mreži na izravan ili neizravan način. Samim time ova topologija (koncept) može živjeti u bilo kojoj drugoj topologiji koja omogućava navedenu funkcionalnost.

Bitno je naglasiti da postoje razlike između fizičke i logičke topologije. Logička topologija prikazuje samo tlocrt putanje podataka između čvorova mreže, ali ne i njenu fizičku (stvarnu) infrastrukturu. Takve topologije najčešće su povezane s načinom na koji se pristupa domaćinima unutar mreže. Odličan primjer logičke topologije je spomenuti P2P model komunikacije koji je zasnovan na potpunoj mrežastoj logičkoj topologiji, dok fizička topologija može biti definirana kombinacijom bilo kojih od gore spomenutih topologija.

Zadnji kriterij podjela mreže je prema njenoj opsežnosti u kojoj dominiraju četiri vrste mreža, a to su gradacijski: mreže osobnih prostora, mreže lokalnog dosega (od jedne institucije), mreže srednjeg dosega (gradske) i mreže globalnog dosega.

Mreže osobnog prostora, pico mreže (*engl. Piconet*) ili PAN (*engl. Personal Area Network*) su sinonimi koji označavaju mreže koje povezuju komponente jednog sustava bežičnim vezama u radijusu do nekoliko metara. Najčešći oblik ovakve mreže u modernome svijetu jesu skupovi uređaja koji razmjenjuju informacije putem Bluetooth

[5] Slika preuzeta sa: https://kids.kiddle.co/Network_topology

tehnologije (bežični zvučnici, bežične slušalice, bežična računalna periferija) ili Infracrvenim valovima (upravljač za televizor).

Mreže lokalnog prostora, lokalne mreže ili engleskim akronimom LAN (*engl. Local Area Network*) su mreže čija lokalnost, odnosno ukupna dužina spojenih kablova mreže, iznosi par kilometara, iako kod nekih mreža lokalnog tipa može biti i veća. Maksimalan broj čvorova u takvim mrežama kreće se od nekoliko stotina do tisuću. Najuspješnija implementacija ovog tipa mreže je lokalna mreža *Ethernet* koja je razvijena 1970-ih godina u istraživačkom centru *Xerox Palo Alto Research Center* (*PARC*). Osnovno načelo rada Etherneta je da svi čvorovi šalju i primaju podatke preko iste veze (topologija sabirnice). Domaćin koji šalje podatke na određenu destinaciju (čvor) u trenutku kada je sabirnica slobodna, odnosno njom ne kola nikakav promet. Prilikom slanja, podaci stižu na sva računala na mreži, ali ako nisu adresirana na njihovu adresu, iste podatke ignoriraju. Podatkovna jedinica u mreži naziva se okvir i pored sadržaja (tijela) poruke sadrži dodatne informacije poput adrese odredišta, adrese izvora, tipa okvira i slično. Uređaji kojima se računala spajaju na takvu vrstu mreže bave se kodiranjem i dekodiranjem signala koji se prenose mrežom, pakiraju i raspakiravaju okvire i sadrže globalno jedinstvenu MAC adresu, a nazivaju se mrežnim kartica. Sve gore navedene funkcije pripadaju mrežnom sloju računalnih mreža kojeg ću detaljnije objasniti u sljedećem poglavlju.

Gradske mreže, mreže srednjeg dosega ili engleskim akronimom MAN (*engl. Metropolian Area Network*) je mreža čija je lokalnost mnogo veća od LAN-a i ugrubo doseže promjer jednog grada (nekoliko desetaka kilometara).

Mreže širokog (globalnog) **prostora** ili WAN (*engl. Wide Area Connection*) su mreže kod kojih nema geografskih omeđenja prostora na kojem se prostiru, tako da se za njih može reći da su mreže neograničene veličine. Najstarija, najveća i najpoznatija mreža takve vrste je Internet.

2.1.3. Slojevi i protokoli mrežnih sustava

Već smo definirali da je cilj računalne mreže osigurati siguran prijenos informacija od jednog do drugog računala. Taj proces jako je izazovan te iziskuje normaliziran odnos

prema podacima i niz logičkih mehanizama koji se bore protiv fizičkih i tehnoloških ograničenja mreže. Kako bi čvorovi u bilo kojoj od tipova mreža opisanih u prethodnom poglavlju mogli sinkronizirano komunicirati, potrebno je kreirati uređeni skup gore navedenih mehanizama i ostalih pravila komunikacije koju će poštivati svi čvorovi mreže.

Takav skup pravila nazivamo protokolom, a niz apstraktnih hijerarhijskih podsustava koji ostvaruju određeni skup operacija unutar jednog čvora nazivamo slojevima mrežnog sustava. Protokoli su obično asocirani s jednim ili više slojeva mrežnog sustava. Primitivna i hipotetska podjela jednog čvora bila bi na fizički sloj koji vrši komunikaciju (emitiranje signala) između računala, translacijski sloj koji niz signala kodira (dekodira), komprimira (dekomprimira) i pretvara u podatak čitljiv čovjeku, te aplikacijski sloj koji s podatkom vrši neke operacije višeg reda, primjerice ispisuje podatak na zaslon računala korisnika.

Kako bi komunikacija između čvorova bila moguća, taj proces (smjer interakcije između slojeva) mora se odvijati u dva različita smjera na dva čvora. Tako, ako domaćin A šalje tekstualnu poruku domaćinu B, komunikacija započinje na aplikacijskom sloju domaćina A uređenog protokolom AP (aplikacijski protokol), prenosi se na translacijski sloj domaćina A, koji na izvorni podatak dodaje (inkapsulira) dodatne podatke o načinu kodiranja i kompresije teksta po pravilima protokola TP (translacijski protokol) te modificiranu strukturu podataka, koju možemo hipotetski nazvati translacijskim paketom, zatim se prenosi na fizički sloj domaćina A koji priprema podatke za slanje protokolom FP (fizički protokol). Fizičkom vezom podaci se prenose do fizičke razine domaćina B, te se potom raspakiraju metodama protokola FP, prenose do translacijskog sloja domaćina B koji translacijski paket pretvara u podatak čitljiv čovjeku uklanjanjem (dinkapsulacijom) dodanih podataka o kodiranju i kompresiji, te vršenju radnji definiranih TP protokolom. Podaci se zatim prezentiraju u aplikacijskom sloju domaćina B po protokolu AP. Naravno, u praksi se nailazi na više posrednih slojeva koji vrše razne funkcije poput segmentiranja podataka, otkrivanja i otklanjanja pogrešaka pri slanju, odnosno primanju podatkovnih jedinica, šifriranja (dešifriranja), komprimiranja (dekomprimiranja), kodiranja (dekodiranja) i ostalih zadaća, no više o tome u nastavku teksta.

U nastavku ćemo se upoznati sa slojevima dvaju najpoznatijih i najkorištenijih modela računalnih sustava i protokola koje koriste, a to su OSI model i model mreže Internet (kojem ću dati veći naglasak).

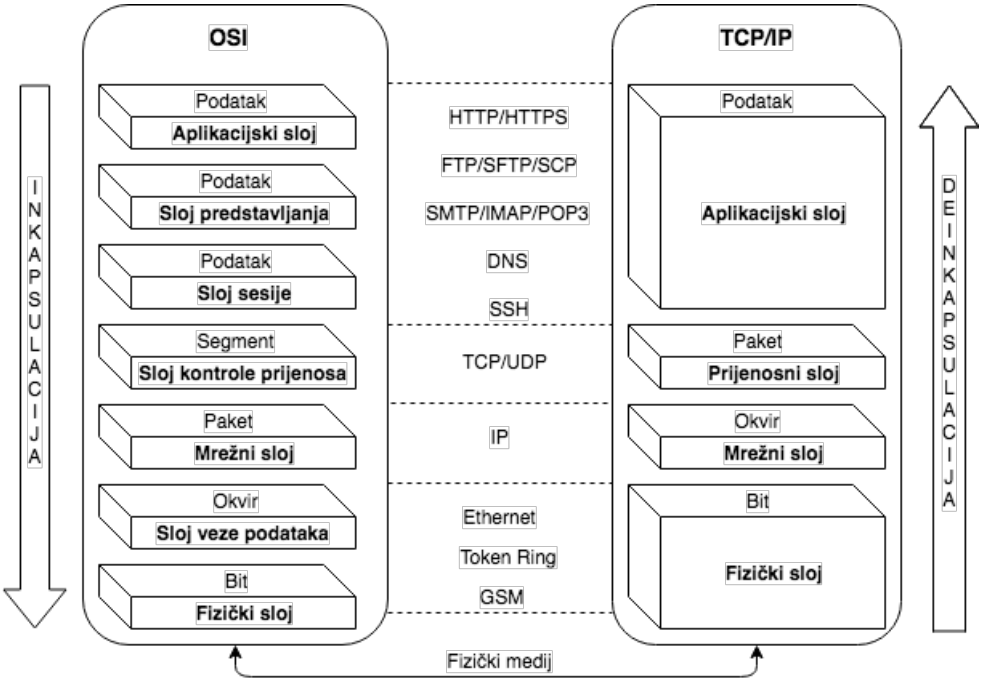
Prvi na repertoaru stoji OSI (*engl. Open System Interconnection*) model koji je razvila radna grupa ISO (*engl. International Standards Organization*) organizacije koja se bavi definiranjem i razvojem standarda za skoro sve vrste proizvoda. OSI model sastoji se od sedam slojeva, pri čemu jedan sloj može sadržavati jedan ili više protokola. Bitno je naglasiti da po sedam slojeva imaju domaćini, dok prijenosnici imaju samo tri sloja, jer su preostala četiri vezana uz rad aplikacija i rade isključivo na domaćinima. Fizički sloj (*engl. Physicial layer*) je sustav „hardverske" razine koji ostvaruje prijenos nizova signala (bitova) između dvaju čvorova. Prijenos bitova mora biti dobro strukturiran i uređen. Četiri najkorištenije takve strukture su okviri, paketi, segmenti te na najnižoj razini bitovi, a možemo ih zamisliti kao kodirane komade izvornog sadržaja koji dolazi za aplikacije razine. Slaganje bitova u okvire izvodi se na sloju (veze) podataka (*engl. Data link*), kroz hardverski uređaj mrežne kartice. Nadalje, zadatak mrežnog sloja (*engl. Network layer*) je da usmjerava pakete (još jedan od oblika strukturiranih nizova bitova) ili okvire koji se prenose odnosno određuje njihov put kroz mrežu (od izvora do odredišta). Jedan od najvažniji slojeva kako OSI-modela tako i Internet-modela je sloj kontrole prijenosa (*engl. Transport Control Layer*). Zadaća ovoga sloja je da, ovisno o protokolu kojeg implementira, osigurava točnost prijenosa podataka, tako da na svaki pripremljeni okvir inkapsulira dodatne podatke, na temelju kojeg druga strana može vrlo brzo detektirati je li se dogodilo bilo kakvo iskrivljenje signala u prijenosu, te od pošiljatelja može ponovno zatražiti okvir. Sloj sesije (*engl. Session Layer*) i sloj predstavljanja (*engl. Presentation Layer*) najčešće rješavaju probleme kodiranje, dekodiranja, šifriranja i dešifriranja sadržaja, odnosno pripreme sadržaja koji se korisniku predstavlja na zaslonu računala. Zadnji i najvažniji sloj za ovaj rad je aplikacijski sloj (*engl. Application Layer*) u kojem krajnji korisnici vrše direktnu interakciju s računalom i njegovim sadržajem putem aplikacija koje žive na operacijskom sustavu računala. Primjer jedne interakcije na aplikacijskom sloju je pretraživanje interneta putem Internetskog preglednika.

Računalna mreža *Internet* službeno je nastala 1983. godine iz mreže ARPANET financirane od strane američke agencije ARPA (*engl. Advanced Research Projects Agency*) u svrhu stvaranja decentralizirane mreže za razmjenu informacija vojnih jedinica u slučaju terorističkih napada ili ratnih stanja gdje bi funkcionalnost mreže ostala netaknuta uništenjem jednog od čvorova mreže. Potencijal ove mreže kasnije je iskorišten u znanstvene svrhe umrežavanja američkih sveučilišta. Vrlo brzo, plasiranjem komercijalnih osobnih računala na tržište te razvojem mrežne infrastrukture, sve više računala postajalo je dijelom Interneta. Vrlo važno je napomenuti da je ova mreža nastala prije definicije teoretskog OSI modela i to kroz praksu i konkretne potrebe korisnika, zbog čega je ovaj model danas najkorišteniji i najpouzdaniji model računalne mreže.

Arhitektura (model) Interneta sačinjena je od samo četiri sloja, a to su hijerarhijskim redom: aplikacijski sloj, sloj kontrole prijenosa, mrežni sloj te fizički sloj. Kao i u OSI modelu, svakim slojem operira posebni protokol, a budući da različiti protokoli obavljaju različite radnje, kombinacija protokola po slojevima nije fiksna. Međutim, tri od četiri sloja većinom koristi iste protokole. Sloj na kojem protokoli često variraju ovisno o korisničkoj potrebi je aplikacijski sloj čije ću protokole ukratko razložiti u sljedećem potpoglavlju.

Arhitektura Interneta često se naziva i TCP/IP arhitektura jer sloj kontrole prijenosa i mrežni sloj najčešće koriste kombinaciju tih dvaju protokola. TCP (*engl. Transmission Control Protocol*) djeluje na sloju kontrole prijenosa i zadužen je za pouzdan prijenos sadržaja. To čini na način da kod nastanka greške u prijenosu ne potvrđuje primitak onih paketa čiji su sadržaji iskrivljeni ili izgubljeni u procesu prijenosa. Drugi protokol koji se često nalazi na istom sloju naziva se UDP (*engl. User Datagram Protocol*) koji funkcionira na malo drugačiji način. UDP za razliku od TCP-a ne ispravlja greške u prijenosu i ne koristi mehanizam potvrde primitaka paketa, čime ne osigurana pouzdan prijenos sadržaja, ali s druge strane omogućava brži prijenos. Ovaj protokol koristan je u situacijama gdje sadržaj svakog paketa ne mora biti potpuno točan, primjerice, pri prijenosu video ili audio sadržaja u realnom vremenu. IP (*engl. Internet Protocol*) je središnji element mrežnog sloja i modela mreže, općenito, koji definira strukturu paketa podataka koji se prenose mrežnom,

adresni prostor (način adresiranja čvorova) u kojem se paketi kreću, i način prenošenja paketa od izvora do odredišta.



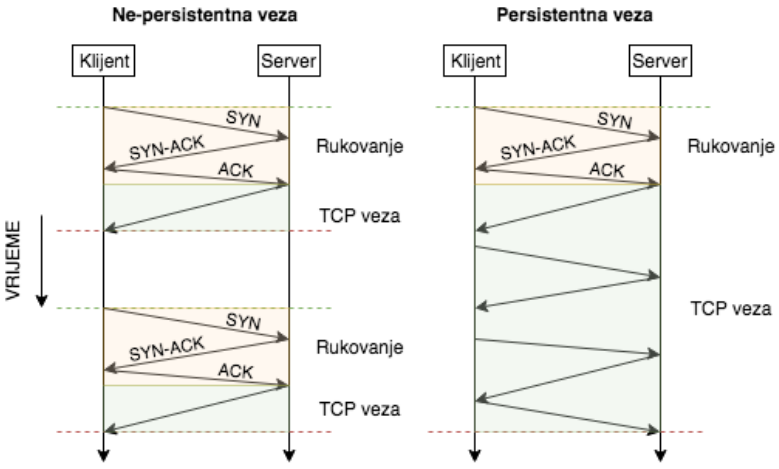
Slika 8. Slojevi, protokoli i jedinice podataka OSI i TCP/IP modela

2.1.4. Aplikacijski protokoli mreže Internet

HTTP (*engl. HyperText Transfer Protocol*) je središnji protokol mrežnog aplikacijskog sustava koji je realiziran kroz dva podsustava, a nazivaju se klijentom i poslužiteljem. Njegova zadaća, kao i svakog drugog protokola, je da definira način na koji klijenti i mrežni poslužitelj razmjenjuju poruke, odnosno način na koji mrežni klijent (primjerice Internetski preglednik na osobnom računalu ili mobilna aplikacija na mobilnom uređaju) potražuje mrežne stranice mrežnog poslužitelja i na koji način poslužitelj dostavlja sadržaje mrežnih stranica klijentu. U trenutku kada korisnik pregledniku zada URL (*engl. Uniform resource locator*) odnosno putanju prema resursu mrežne stranice, preglednik upućuje HTTP zahtjev odgovarajućeg sadržaja (poruke) mrežnom poslužitelju koji isti zahtjev prima i na njega odgovara relevantnim mrežnim sadržajem tj. objektom koji

klijentska aplikacija interpretira na neki programirani način (primjerice renderira mrežnu stranicu na zaslonu korisnika). Mrežnim objektom ili resursom naziva se svaka datoteka koja sadrži neki HTML (*engl. Hypertext Markup Language*) kodni zapis od kakvih se tvore mrežne stranice, a mogu sadržavati i digitalne zapise slike, videa i zvuka.

Proces komunikacije između klijenta i poslužitelja možemo podijeliti u dva dijela, a to su proces uspostavljanja TCP veze i proces razmjene sadržaja. Postupak uspostavljanja TCP veze naziva se rukovanjem u tri koraka (*engl. Three-way handshake*). Prvi od tri koraka uspostavljanja veze započinje slanjem jednog sinkronizacijskog upravljačkog TCP segmenta (SYN) poslužitelju koji potom potvrđuje sinkronizaciju slanjem svog upravljačkog TCP segmenta (SYN-ACK) klijentu. U konačnici klijent šalje odgovor (ACK) poslužitelju kojim potvrđuje TCP vezu i traži sadržaj slanjem posebno definirane HTTP poruke na koju server odgovara odgovarajućim mrežnim objektom (razmjena sadržaja). Kada govorimo o TCP vezi između klijenta i poslužitelja, ona može biti ne-perzistentna i perzistentna. Kod ne-perzistentne veze, poslužitelj će pokrenuti proces prekidanja veze nakon što je poslao traženi sadržaj. S druge strane, perzistentna veza ostaje otvorena sve dok traje jedna komunikacijska sesija što klijentu omogućuje da preko iste veze od server traži veći broj objekata. Ovaj tip komunikacije omogućuje ostvarivanje DoS podtipa napada na poslužitelje o kojem ću više govoriti u trećem i petom poglavlju. Shema komunikacije između klijenta i poslužitelja korištenjem HTTP protokola ilustrirana je na slici 9.



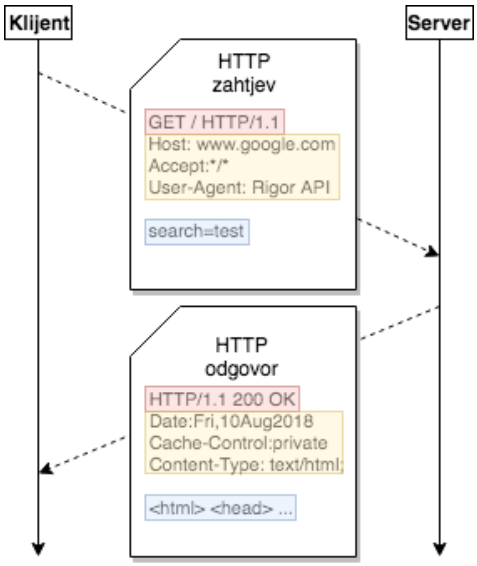
Slika 9. Ilustracija uspostave i održavanje dvaju vrsta TCP veza

HTTP poruke dijele se u dvije osnovne grupe: poruke-zahtjevi (klijentska strana) i poruke-odgovori (poslužiteljska strana). Poruka-zahtjev klijenta sadrži više redaka od kojih se prva naziva retkom zahtjeva (*engl. Request line*), dok se ostali redovi nazivaju redovima zaglavlja (*engl. Header line*). Redak zahtjeva se sastoji od tri elementa, a to su redom metoda, URL te verzija HTTP protokola. Metode podržane HTTP standardom su: „GET“, „POST“, „HEAD“, „PUT“, „PATCH“ i „DELETE“ te u suštini predstavljaju glagol odnosno radnju na engleskom jeziku koja se nad nekim resursom treba izvršiti. Tako primjerice neki sadržaj možemo dohvatiti, kreirati, modificirati ili izbrisati. U redovima zaglavlja klijent može definirati niz atributa specifičnih za neki zahtjev i poput jezika na kojima sadržaj traži, način kodiranja sadržaja, podatke o klijentu poput operacijskog sustava i platforme kako bi od poslužitelja dobio personalizirani sadržaj i slične, autentikacijske podatke, kolačiće i slično. Također, klijent u svom zahtjevu može poslati i tijelo (*engl. Body*) odnosno sadržaj koji će server iskoristiti primjerice pri kreiranju novog resursa. Poruka-odgovor poslužitelja sastoji se od sličnih dijelova, a to su: statusni redak, redovi zaglavlja i tijelo odgovora. Statusni redak također se sastoji od tri elementa od kojih prvi označava verziju HTTP protokola, kodnu oznaku stanja te tekstualnu poruku koja opisuje prethodno kodnu oznaku stanja. Primjere nekih od predefiniranih kodnih oznaka možemo vidjeti u tablici 1.

Statusni kod	200	301	302	303	403	404	500	503
Opis	OK	Moved Permanently	Moved Temporarily	See Other	Forbidden	Not Found	Server Error	Service unavailable

Tablica 1. Prikaz najfrekventnijih HTTP statusnih kodova

Redovi zaglavlja sadrže metapodatke o domaćinu i poslužitelju te sadržaju poput informacija o tome kada je resurs zadnji put modificiran, kolika je dužina resursa, tip resursa, podatak o tome hoće li poslužitelj pokrenuti proces prekida TCP veze nakon slanja odgovora i slično. Zadnji dio sadrži sam sadržaj resursa koji može biti i prazan. Primjer razmjene HTTP poruka između klijenta i poslužitelja možemo vidjeti na slici 10.



Slika 10. Primjer i segmenti HTTP zahtjeva i odgovora

Problem HTTP komunikacije je što pri razmjeni osjetljivog sadržaja ili personaliziranog (osobnog) sadržaja podaci u prijenosu ostaju nezaštićeni, odnosno nisu zakriveni (šifrirani). Tako da pri prijenosu podataka kroz mrežu, svaki čvor koji posreduje u prijenosu podatkovnih jedinica, može imati kompletan uvid u sadržaj svakog zahtjeva, koji se, ako se koristi na zlonamjeran način, smatra hakerskim napadom (više u trećem poglavlju).

Kao rješenje navedenog problema formiran je novi protokol koji danas čini standard sigurne komunikacije putem Interneta pod akronimom HTTPS (*engl. HyperText Transfer Protocol Secure*). Ovaj je protokol kombinacija HTTP protokola i SSL/TLS protokola. SSL/TLS također je aplikacijski protokol nastao kombinacijom starijeg SSL (*engl. Secure Sockets Layer*) i novijeg TLS (*engl. Transpot Layer Security*) protokola koji u suštini omogućuje sigurnu komunikaciju putem Interneta korištenjem kriptografskih algoritama čiji se temelj zasniva na kriptografiji javnog ključa o kojem ću reći nešto više u sljedećem potpoglavlju. Ovaj protokol najčešće je korišten kao dodatni sloj povrh tradicionalnih protokola za razmjenu podataka na aplikacijskom sloju.

FTP (*engl. File Transfer Protocol*) je protokol aplikacijske razine koji ima specifičnu primjenu u prijenosu datoteka između dvaju domaćina. Način na koji FTP funkcionira je da korisnik putem korisničkog imena i lozinke pristupa datotečnom sustavu

drugog udaljenog domaćina. Sukladno korisničkim pravima ima pristup skupu datotečnog prostora. FTP kao i HTTP koristi TCP vezu pri komunikaciji između domaćina, pri čemu se kreiraju dvije paralelne veze; jedna upravljačka veza kojom se održava otvorena sesija, vrši autentikacija i autorizacija korisnika, vrši navigacija nad direktorijima te podatkovna veza putem koje se vrše prijenosi podataka. Kao i kod HTTP-a, komunikacija između domaćina ostaje nezaštićena iz čega nastaje SFTP (*engl. Secure File Transfer Protocol*) koji je, kao i u prethodnom slučaju, kombinacija nativnog protokola i SSL/TLS protokola.

Skup protokola razvijen je za potrebe razmjene elektronskog sadržaja korištenjem sustava računalne (elektroničke) pošte, a to su: SMTP (*engl. Simple Mail Transfer Protocol*), POP3 (*engl. Post Office Protocol*) i IMAP (*engl. Internet Message Access Protocol*). Sustavi računalne pošte pružaju usluge asinkrone komunikacije što govori da primatelj poruke ne mora primati njen sadržaj u isto vrijeme kada pošiljatelj proizvodi taj isti sadržaj. Svaki od navedenih protokola služi istoj svrsi, ali svoj posao obavljaju na malo različite načine. Primjerice POP protokol pohranjuje i briše poruke elektroničke pošte s poslužitelja na lokalno računalo, dok IMAP ostavlja poruke na serveru i korisniku prikazuje privremenu kopiju. Tako da pri brisanju lokalne kopije poruke koristeći POP protokol gubimo opciju oporavka datoteke s poslužitelja kako je to slučaj s IMAP protokolom. Prednost POP protokola je što sadržaju (posljednje verzije) elektroničkog sandučića možemo pristupiti i kada nemamo pristup Internetu. Svi protokoli također koriste TCP veze između klijenta i poslužitelja (centralnog pohranioca elektroničke pošte) i najjednostavnije rečeno koriste kombinirani pristup razmjene sadržaja HTTP-a i FTP-a. Ovisno o protokolu, klijent šalje posebno strukturiranu poruku kojom se predstavlja s korisničkim imenom i lozinkom, nakon čega se uspostavlja TCP veza putem koje se ponovno posebno strukturiranim porukama pristupa i manipulira privatnim poštanskim pretincem.

U prethodnim poglavljima spominjali smo da su sva računala prisutna u mreži označena jedinstvenom fizičkom MAC, odnosno logičkom IP adresom. Takva nomenklatura čovjeku nije praktična te se za potrebe pristupanja mrežnim aplikacijama koriste spomenuti URL-ovi, odnosno adrese poslužitelja definirane prirodnim jezikom. Za potrebe prevođenja URL adresa u strojno čitljive adrese računala unutar mreže na Internetu

koristi se DNS (*engl. Domain Name Server*). DNS je distribuirana baza podataka, koja je implementirana pomoću jednog globalnog sustava (DNS poslužitelja) koji tvore hijerarhijsku stablastu strukturu statičnih (latentno nepromjenjivih) IP adresa te njihovih odgovarajućih URL adresa. Tako da pri svakom pristupu nekoj od Internetskih domena, klijent prije uspostave TCP veze sa serverom, mora od DNS poslužitelja DNS protokolom dobiti informaciju o IP adresi udaljenog poslužitelja. DNS se svakog trenutka ažurira i distribuirano razmjenjuje informacije sa svim ostalim DNS serverima diljem svijeta.

Važno je napomenuti da jedan server može posluživati više servisa (tipova mrežnih aplikacija) odnosno operirati s više protokola, jer u suštini svaki protokol odgovara jednoj vrsti servisa. Iz tog razloga, svaka mrežna aplikacija ima svoj broj ulaznih vrata (*engl. Port Number*). Primjerice, mrežni poslužitelj kao aplikacija, označen je brojem vrata 80 dok poslužitelj za računalnu poštu, kao dio aplikacije računalne pošte, koristi vrata broj 25. Popis vrata prema odgovarajućim protokolima predočen je na tablici 2. S ovim paragrafom došli smo i do definicije mrežnih aplikacija, a mogli bismo jednostavno reći da su to specifične usluge koje poslužitelj na domaćinu ustupljuje klijentima, putem otvorenih vrata i protokola, koristeći Internet ili bilo koju drugu računalnu mrežu.

Protokol	HTTP(S)	FTP	SMTP	IMAP(S)	POP3(S)	DNS	SSH
Vrata	80 (433)	20/21	25	143 (993)	110 (995)	53	22

Tablica 2. Pregled nekih od aplikacijskih protokola i odgovarajućih brojeva vrata

Na aplikacijskom sloju još djeluje veliki broj protokola koje, za potrebe ovog rada i provedbe penetracijskog testiranja u petom poglavlju, nije potrebno detaljno razlagati.

2.1.5. Sigurnost i zaštita

U računalnoj mreži treba štititi mrežu kao tehnološki sustav, komunikatore i informacije sadržaja. U govoru o mrežnoj komunikaciji ističu se tri osnovna aspekta zaštite i sigurnosti: zaštita o povjerljivosti sadržaja, zaštita integriteta (izvornosti) sadržaja od njegovog zlonamjernog iskrivljenja u prijenosu i utvrđivanje autentičnosti komunikatora

(Radovan, 2011). Ova tri aspekta zaštite i sigurnosti aplikacija mogu se odvijati na više slojeva mreže različitim softverskim i hardverskim mehanizmima. U našem kontekstu pozabavit ćemo se aplikacijskim slojem i definiranjem ključnih kriptografskih algoritama u ostvarivanju sigurne komunikacije.

Prije definiranja nama važnog koncepta javnog ključa i algoritama simetričnog i asimetričnog šifriranja, parafrazirat ću neke osnovne pojmove kriptografije prema Dujelli i Maretiću (2007). Kriptografija je znanstvena disciplina koja se bavi proučavanjem metoda za slanje poruka u takvom obliku da ih samo onaj kome su namijenjene može pročitati. Poruku koju pošiljalac želi poslati primaocu zovemo otvorenim tekstom (*engl. Plaintext*). Postupak kojim pošiljalac transformira otvoreni tekst koristeći unaprijed dogovoreni ključ naziva se šifriranjem, a dobiveni rezultat šifrantom (*engl. Ciphertext*) ili kriptogramom. Kriptografski algoritam ili šifra je matematička funkcija koja se koristi za šifriranje (kriptiranje) i dešifriranje (dekriptiranje). Funkcije se biraju iz određene skupine funkcija u ovisnosti o ključu, a skup svih mogućih vrijednosti ključeva nazivamo prostorom ključeva.

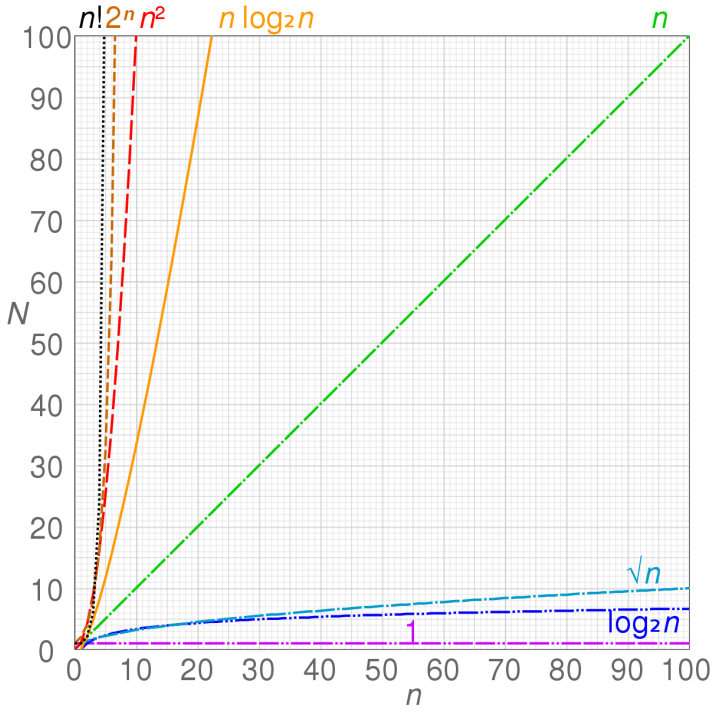
Kriptosustav se sastoji od kriptografskog algoritma, te svih mogućih otvorenih tekstova, šifranata i ključeva. Kriptosustave obično klasificiramo s obzirom na sljedeća tri kriterija: tip operacije koje se koriste pri šifriranju, način na koji se obrađuje otvoreni tekst, tajnost i javnost ključeva. U prvom kriteriju razlikujemo supstitucijske šifre kojima se svaki element otvorenog teksta zamjenjuje nekim drugim elementom, transpozicijske šifre u kojima se elementi otvorenog teksta permutiraju (premještaju), te hibridni sustavi koji kombiniraju dvije navedene metode šifriranja. U drugoj podjeli razlikujemo blokovne šifre, kod kojih se obrađuje jedan po jedan blok elemenata otvorenog teksta koristeći jedan te isti ključ K, te protočne šifre (*engl. Stream cipher*) kod koji se elementi otvorenog teksta obrađuju jedan po jedan koristeći pritom niz ključeva (*engl. Keystream*) koji se paralelno generira.

Prema tajnosti i javnosti ključeva kriptosustave možemo podijeliti na simetrične i asimetrične (kriptosustave s javnim ključem). Simetrični sustavi omogućavaju šifriranje (enkripciju) i dešifriranje (dekripciju) otvorenog teksta korištenjem jednog zajedničkog dijeljenog ključa, što znači da se dešifriranje može izračunati poznavajući ključ šifriranja i

obratno. Sigurnost ovoga sustava leži isključivo u tajnosti ključa. Kod asimetričnih kriptosustava ključ za dešifriranje nije moguće izračunati poznavanjem ključa za šifriranje u razumnom vremenu. U ovom sustavu postoje dva para privatnih (tajnih) i javnih ključeva, gdje jedan par pripada jednom komunikatoru, a drugi par drugome. Svi javni ključevi dostupni su svima i služe za šifriranje otvorenih poruka koje će biti upućene vlasniku javnog ključa. Čar ovog sustava je što samo korisnik koji posjeduje drugu polovinu para ključeva (privatni ključ) može dešifrirati šifrant, što znači da se sigurnost ovog sustava ne oslanja na tajnosti ključa jer se razmjena ključa nikad nije dogodila.

Dešifriranje šifranta također je moguće i u obrnutom smjeru. Ako komunikator šifrira neku poruku svojim privatnim ključem, bilo tko može dešifrirati šifrant javnim ključem. Ovaj slučaj stvorio je mogućnost jedinstvenog prepoznavanja komunikatora, odnosno stvaranje digitalnog potpisa, jer ako se dešifriranjem šifranta javnim ključem dobio očekivani otvoreni tekst poput imena, prezimena i slično znači da je poruka morala doći od komunikatora koji posjeduje jedinstveni privatni ključ. Ideju javnog ključa prvi su javno iznijeli Whitfield Diffie i Martin Hellman 1976. godine, kada su dali prijedlog rješenja problema razmjenjivanja ključeva za simetrične kriptosustave putem nesigurnih komunikacijskih kanala (Diffie-Hellman razmjena ključeva) koji danas koristi HTTPS i mnogi drugi enkripcijom poduprti protokoli. Moć nereverzibilnog izračuna između privatnog i javnog ključa krije se u teoriji brojeva i modularnoj aritmetici točnije u teškoći faktorizacije velikih prirodnih brojeva. Uzimo jednostavan primjer i pomnožimo dva četveroznamenkasta broja 1234 i 5678. Množenje je radnja čija je vremenska kompleksnost izračuna između linearne (n) i polilogaritamske ($n\ log_2 n$), odnosno linearno ili polilogaritamski raste s većim brojevima. Potom dobiveni umnožak 7006652 pokušajmo rastaviti na faktore. Kako čovjeku, tako i računalu ova zadaća postaje drastično teža što se radi o većem broju, odnosno kompleksnost izračuna ove funkcije je brzorastuća, u ovom slučaju polinomijalna (n^m). Vremenska kompleksnost nekog problema ne ovisi samo o prirodi problema već i algoritmu koji implementira rješenje, tako često različiti algoritmi koji rješavaju isti problem imaju različitu vremensku kompleksnost. Time smo dobili svojstvo gdje se izračun, primjerice generiranje javnog i privatnog ključa ili šifriranje poruke odvija brzo u jednom smjeru, a (gotovo) nemoguće u drugom smjeru. Kada kažemo

„gotovo nemoguće“, mislimo da je nemoguće dobiti rezultat u razumnom vremenu (sekunde, sati, dani, tjedni, mjeseci, godine), većinom su u pitanju desetljeća ili stoljeća. Iza ove premise funkcioniraju moderni asimetrični algoritmi čiji opis nadilazi potrebe ovoga rada. Usporedbu krivulje polinomijalne i linearne funkcije možemo vidjeti na slici 11.



Slika 11. Usporedba krivulja vremenske kompleksnosti[6]

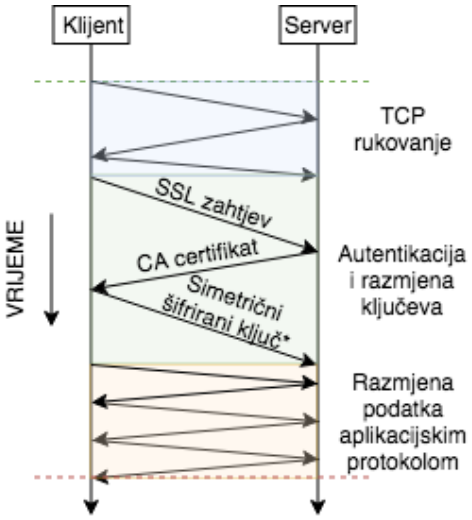
RSA je najstariji, najpoznatiji i najkorišteniji sustav za šifriranje uporabom javnog ključa i koristi gore navedenu implementaciju kriptosustava. Najveći nedostatak šifriranja asimetričnim kriptosustavom je brzina. U odnosu na najpoznatiji sustav sa simetričnim ključem DES (*engl. Data encryption standard*), RSA izvodi šifriranje između 100 i 10000 puta sporije ovisno implementira li se DES hardverski ili softverski. U modernoj praksi nailazimo na razna primijenjena matematička rješenja poput eliptičnih krivulja koja pokušavaju optimizirati i unaprijediti sustave za asimetrično šifriranje. Neki od korištenih sustava su: DSA, EIGamal, ECDH, ECDSA, EdDSA.

[6] https://en.wikipedia.org/wiki/Time_complexity

Uočimo da, kombiniranjem simetričnog i asimetričnog sustava šifriranja, možemo otkloniti negativne strane i jednog i drugog sustava. U takvoj kombinaciji, za generiranje privatnog i javnog ključa te sigurnu razmjenu jednog simetričnog ključa preko mreže, koristi se RSA, dok se za daljnju komunikaciju (šifriranje) koristi daleko brži simetrični DES ili AES (*engl. Advanced encryption standard*) sustav. Time u niti jednom trenutku simetrični ključ nije bio izložen u otvorenom tekstu.

No, postavlja se pitanje kako znamo da komuniciramo s točnim krajnjim komunikatorom, a ne s nekim od posrednika (mrežnih čvorova) koji se može lažno predstaviti (lažirati digitalni potpis) pošiljatelju poruke i poslati mu vlastiti javni ključ umjesto javnog ključa krajnjeg komunikatora, čime narušava cjelokupni integritet komunikacije. Kako bi se spriječio ovaj problem, potrebno je pravno ovjeriti (utvrditi) i upariti fizičke osobe s njihovim svojstvima (javnim ključevima), kako bi sa sigurnošću mogli znati da komuniciramo s odgovarajućim komunikatorom. Takve poslove izvodi institucija koju u otvorenom prijevodu možemo nazvati ovjerovnom službom (*engl. Certification Authority - CA*).

Ovjerovna služba izdaje potvrdu (certifikate) s kojom se potvrđuje da je dani entitet (osoba, tvrtka, usmjerivač, server) zaista onaj koji taj entitet tvrdi da jest. Identifikacija entiteta treba biti jednoznačna na globalnoj razini i može uključivati ime, prezime, OIB te niz drugih podataka. Potvrda koju izdaje ovjerovni ured zapisana je u digitalnom obliku i digitalno je potpisana s tajnim ključem ovjerovne službe. No, ako ovjerovna služba ovjerava certifikat svojim digitalnim potpisom, kako znamo da možemo vjerovati ovjerovnoj službi? Isti postupak certificiranja ponavlja se dok se ne dođe do vrhovne ovjerovne službe poput vlade, države ili još većih organizacija. Time se zapravo gradi jedan veliki lanac povjerenja koji se oslanja na ljudsko poštenje, što stvara novi stari problem ljudskog društva, a to je korumpiranost i nemoral. Ilustraciju Diffie-Hellman razmjene ključeva i uspostave sigurne komunikacije putem SSL/TLS protokola u računalnoj mreži možemo vidjeti na slici 12 (primijetimo da dodani TLS sloj produljuje vrijeme uspostavljanja veze s poslužiteljom).



Slika 12. Diffe-Hellman razmjena ključeva i uspostavljanje SSL/TLS veze
(* simetrični ključ šifriran je javnim ključem dobivenim iz certifikata koji server dešifrira i raspakira svojim privatnim ključem)

Važno je spomenuti još jedan način uspostavljanja sustava zaštite u računalnoj mreži, koji se zasniva na ograničavanju pristupa pojedinim mrežama i na ograničavanju prijenosa pojedinih vrsta sadržaja u pojedine mreže i iz tih mreža. Postoji više sredstava za uspostavu ove vrste zaštite, a to su: vatreni zidovi ili vatrozidovi (*engl. Firewalls*), sustavi za otkrivanje napada (*engl. Intrusion detection systems - IDSs*) i sustavi za sprječavanje upada (*engl. Instrusion prevention systems - IPSs*). Nama najvažniji sustav je softverski sustav vatrozid koji se obično postavlja na računalo koje ima ulogu vrata koja povezuju računalnu mrežu neke tvrtke ili institucija s globalnom mrežom Internet.

Vatrozid ima ulogu vratara (na tim vratima) koji dopušta neke prijenose podataka (paketa, okvira i slično), a druge prijenose sprječava. Drugim riječima, vatrozid se bavi filtriranjem tokova podataka. Načela prema kojima vatrozidi izvode filtriranje paketa oblikuju se na temelju sljedećih atributa IP paketa (okvira): IP adresa, protokol, ulazna i izlazna vrata, stanje TCP veze (bitovna oznaka zaglavlja) i slično. Kombinacije ovih podataka čine tablicu upravljanja pristupom (*engl. Access control list*) koja koherentno određuje ponašanje vatrozida i dinamički se proširuje, čiji primjer možemo vidjeti u Tablici 3.

Što učiniti?	Mrežna adresa izvora	Mrežna adresa odredišta	Protokol	Vrata izvora	Vrata odredišta	Bitovna oznaka
Propusti	192.94.15/20	192.95.12/20	TCP	>1024	80	ACK
Propusti	192.94.15/20	192.95.12/20	UDP	>1024	53	ACK
Odbaci	192.94.15/20	192.95.12/20	Svi	Svi	Svi	Sve

Tablica 3. Primjer tablice upravljanja pristupom

Rad sustava tipa IDS i IPS zasniva se na otkrivanju anomalija i radi na način da pokušava opisati ono što je normalno i otkrivati ono što odstupa od toga profila. Ovakvi sustavi najčešće promatraju zaglavlja i sadržaje svih IP paketa i TCP segmenata, trajanje TCP veze, učestalost zahtjeva i slično. Također, efikasni su u otkrivanju i prevenciji DoS napada i napada grubom silom, a o tome ću više govoriti u trećem poglavlju.

2.2. Baza podataka

Svaki skup međusobno povezanih (i primjereno oblikovanih) podataka nekog sustava naziva se bazom podataka (Radovan, 1993). Baze podataka su neizostavni dio svakog informacijskog sustava jer omogućuju informacijama (podacima) da u sustavu žive, modificiraju se i umiru na strukturiran način ovisno o čimbenicima i akcijama informacijskog sustava. Kako se u bazama podataka čuvaju razni privatni podaci korisnika informacijskog sustava, time postaju jedne od najvećih meta hakerskih napada. Budući da se u tekstu isprepleću dva pojma koji se naizgled čine kao sinonimi, važno je napraviti distinkciju između podatka i informacije. Podatak je bilo koja činjenica u formaliziranom obliku koja ne posjeduje kontekst, dok je informacija (ili obavijest) podatak od vrijednosti za donošenje odluka, odnosno podatak doveden u kontekst (Tuđman et. al, 1991). Primjerice, zelena boja je podatak dok je zelena boja na semaforu križanja (kontekstu) informacija ili obavijest, jer je to podatak na temelju kojeg donosimo odluku o nastavku vožnje na specifičnom križanju ceste.

Budući da nas zanima uloga i primjena baza podataka u kontekstu mrežnih aplikacija, objasnit ću važnost, koncept i podjelu baza podataka korištenih u izradi

modernih aplikacija. Prema Radovanu (1993) baze podataka nose neke od temeljnih odlika koje pomažu aplikacijama u interakciji s podacima, a to su: kontrola redundancije (zalihost), integritet sustava, korištenje zajedničkih podataka, zaštita podataka, standardizacija podataka te optimizacija cjeline.

Glavna podjela baza podataka je na relacijske i nerelacijske baze podataka. U mrežnim aplikacijama koriste se obje vrste baza, vrlo često u istoj implementaciji aplikacije. Relacijska baza podataka prati jednostavno načelo relacijskog pristupa koje kaže da bi bazu podataka trebao činiti skup tablica podataka koje su logički povezane relacijama, te da se, kao rezultat svake operacije nad sadržajem baze, treba dobiti nova tablica podataka. Ovakav koncept baze podataka zahtijeva unaprijed definiranu shemu podataka (tablica). Uobičajeni jezik za rad s relacijskom bazom podataka je SQL (*engl. Structured Query Language*), što bi u grubom prijevodu akronima značilo *strukturirani jezik za upite*. Budući se SQL upiti na bazu dinamički generiraju na temelju korisnikovih unosa u mrežnoj aplikaciji, svaka neopreznost može kompromitirati podatke. U trećem i petom poglavlju rada govorit ću i demonstrirati najčešći napad u domeni baza podataka, a to je SQL injekcija.

Svakom bazom podataka upravlja sustav za upravljanje bazom podataka (*engl. Database Management System - DBMS*). U praksi poznajemo nekoliko komercijalnih i nekomercijalnih DMBS-ova korištenih u izgradnji mrežnih aplikacija, a to su: MySQL, Microsoft SQL Server, PostgreSQL, Oracle, SQLite. Važno je napomenuti da instance DMBS-a ne moraju biti smještene na domaćinu mrežne aplikacije, što olakšava procese migracije i skaliranja (proširivanja kapaciteta ili performansi) baze podataka.

Pod pojmom nerelacijskih baza podataka, smatramo sve baze koje ne prate navedeno načelo, odnosno čiji se podaci ne zasnivaju na strukturi podataka u obliku tablica te njihovih međusobnih relacija. Ove baze u pravilu koriste strukture podataka poput dokumenata, ključ – vrijednost parove, grafove i slične nestandardne strukture te pri dodavanju novih podataka najčešće ne moraju poštivati nikakvu unaprijed dogovorenu shemu. Najpoznatije nerelacijske baze podataka, odnosno sustavi za upravljanje bazama podataka su: *MongoDB*, *BigTable*, *Redis*, *RavenDB Cassandra*, *HBase*, *Neo4j* i *CouchDB*. Neke od glavnih razlika između ova dva tipa baza podataka su u brzini, skalabilnosti i već

spomenutim strukturama podataka. Neke od nerelacijskih baza podataka žive u radnoj memoriji u kontrastu na ostatak baza čiji podaci perzistiraju u trajnoj memoriji računala, stoga je brzina pristupa između baza usporedna razlici brzina pristupa radne memorije i trajne memorije. Također, nerelacijske baze podataka lakše se skaliraju horizontalno, odnosno na način da se lako mogu dodati nove instance baze, jer podaci ne ovise jedan o drugome, odnosno ne posjeduju relacije. S druge strane, DBMS relacijske baze sa sobom nose dosta kompleksnije tehnološko rješenje koje brine o integritetu i sigurnosti podataka, s toga horizontalno skaliranje postaje teška zadaća, jer podaci između dvije baze moraju biti međusobno povezani. Iz toga, razlika ove baze najlakše se, ali ne i uvijek, skalira vertikalno, povećanjem količine memorije ili snage procesora računala. Na temelju navedenih faktora ovisi i odabir tipa baze podataka, a najčešće ovisi o strukturi i važnosti pohranjivanog podatka (Xplenty, 2017).

U konačnici primijetimo da su i baze podataka u suštini servisi te žive na domaćinima kojima se može pristupati s udaljenih računala. S toga, ovisno o DBMS-u, svako pristupanje bazi podataka iziskuje otvaranje posebnih vrata. Primjerice, *PostgreSQL* koristi vrata broj 5432 dok *MySQL* koristi vrata broj 3306 ili 3307. Ove informacije dosta su nam korisne pri otkrivanju tipa baze podataka odnosno DBMS-a skeniranjem otvorenih vrata domaćina, ako baza živi na istom domaćinu kao i mrežna aplikacija.

3. Sigurnost i sigurnosne ranjivosti mrežnih aplikacija

Ovo poglavlje posvećeno je definiranju sigurnosnih ranjivosti mrežnih aplikacija i aplikacijskog sloja računalnih mreža generalno. Prije svega razjasnit ću distinkciju između autentikacije i autorizacije korisnika te modernih mehanizama implementacije istih. U konačnici opisat ću najčešće i najpoznatije napade prema mrežnim aplikacijama i korisnicima mrežnih aplikacija čije je poznavanje nužno za razumijevanje petog poglavlja.

3.1. Autentikacija i autorizacija

Gotovo svaka mrežna aplikacija koja poslužuje personaliziran sadržaj i verificira identitet korisnika koristi mehanizme autentikacije i autorizacije. Prema Balochu (2015) u kontekstu sigurnosti mrežnih aplikacija, autentikacija se odnosi na postupak verificiranja korisnika i utvrđivanja njegovog identiteta u sustavu. S druge strane, autorizacija je proces koji se javlja nakon autentikacije i definira prava pristupa nekom sadržaju ili funkcijama sustava. Autorizacija korisnika najčešće se bazira na njegovim atributima i povijesti interakcija s aplikacijom. Primjerice, u mrežnoj aplikaciji koja ima osobine foruma, korisnik koji nije kreirao raspravu ne može brisati sadržaj rasprave, već to samo može raditi kreator. S druge strane, korisnik koji je objavio neki odgovor u raspravi može modificirati svoj odgovor, dok istu radnju ne može obaviti nitko pa čak ni kreator rasprave. Svi ti korisnici ne bi mogli obavljati niti jednu od navedenih funkciju ako u početku nisu autenticirani kao legitimni korisnici sustava. Ako se ijedna od navedenih hipotetskih pravila prekrši, govorimo o neovlaštenom ili neautoriziranom pristupu mrežnoj aplikaciji. Autorizacija se uglavnom definira u programskom kodu mrežne aplikacije, a propusti u logici implementacije dovode do neautoriziranog pristupa.

U praksi postoje razni mehanizmi autentikacije koji su kroz povijest evoluirali. Najpopularniji pristup autentikaciji je korištenje tajnih zaporki ili kombinacija javnih i tajnih informacija, primjerice adresa elektronske pošte i korisnička zaporka. Ovaj mehanizam autentikacije zasniva se u tajnosti informacije koju može znati jedino validan

korisnik, odnosno faktor sigurnosti ovakvog način autentikacije zasniva se na nečemu što korisnik zna. Neki moderniji mehanizmi autentikacije koriste i druge faktore poput nečega što korisnik posjeduje ili nečega što korisnik je. Tako u prvom slučaju korisnik može registrirati neki uređaj (najčešće mobilni) koji će služiti kao potvrda o autentikaciji. Drugi i ujedno najlegitimniji oblik autentikacije je biometrijski postupak koji verificira korisnika na temelju fizičkih osobina koje ga definiraju. Nedostatak biometrijskog pristupa je kompleksnost i cijena implementacije koja u modernije vrijeme postaje sve manja te se implementira čak i na mobilnim uređajima (primjerice čitači otiska prstiju ili skeniranje rožnice oka) (Fettinger, 2018).

No postupak autentikacije ne bi trebao biti limitiran na samo jedan faktor sigurnosti, već koristiti kombinaciju gore navedenih faktora. Tako se danas u većini popularnih aplikacija vrši višefaktorska autentikacija (*engl. Multifactor authentication - MFA*). Primjer ovog hibridnog pristupa bio bi korištenje mobilnog uređaja kao nečeg što posjedujemo i korisničkih zaporki kao nečeg što znamo. Tako sustav u slučaju pristupa putem korisničke zaporke s dosad nepotvrđene lokacije, primjerice drugog grada ili države od korisnika može zatražiti potvrdu pristupa putem mobilnog uređaja kojeg posjeduje. Ovim mehanizmom sprječava se neovlaštena autentikacija ukoliko je jedan od faktora postao kompromitiran. Također, ovim načinom samo osoba koja posjeduje sve faktore autentikacije može uistinu potvrditi da je osoba za koju tvrdi da jest (Fettinger, 2018).

U slučaju interaktivnih mrežnih aplikacija nepraktično je i nepoželjno korisnika autenticirati prilikom svake HTTP akcije. S toga se autentikacija korisnika obavlja na jednom mjestu prilikom pristupa sustavu te različitim metodama pamti njegov identitet korištenjem sesija (*engl. Session*). Mrežna aplikacija nakon uspješne autentikacije stvara neki oblik podatka šifriran tajnim ključem (najčešće simetričnim) kojeg dodjeljuje korisniku i kojeg korisnik čuva te šalje svakim novim zahtjevom na mrežnu aplikaciju. Primjer takvih podataka su pristupne značke (*engl. Access token*), API ključevi (*engl. API keys*) i kolačići (*engl. Cookies*). Gotovo svaki od navedenih struktura sa sobom nosi niz metapodataka koji definiranju atribute sesije poput vremena trajanja sesije. Budući da korisnik navedene podatke čuva lokalno i šalje prilikom svakog zahtjeva, svako presretanje

ili korištenje nezaštićene veze kompromitiralo bi sesiju korisnika te stvorilo prostor za ostvarivanje neovlaštenog pristupa mrežnoj aplikaciji kroz prizmu drugog korisnika.

3.2. Sigurnosne ranjivosti

U narednom tekstu opisat ću neke od najčešćih sigurnosnih ranjivosti mrežnih aplikacija prema OWASP-u (2017). Budući da napada ima mnogo, opisat ću one koje je potrebno znati za razumijevanje provedenih napada u petom poglavlju.

3.2.1. XSS ranjivost

XSS (*engl. Cross site scripting*) jedna je od najpopularnijih ranjivosti mrežnih aplikacija koja se veže uz povijest gotovo svake popularne mrežne stranice. Ova ranjivost zasniva se na injektiranju koda u obliku malicioznih skripti klijentskog programskog jezika (najčešće *JavaScripta*) koji je namijenjen da se izvrši na klijentskim računalima korisnika mrežne aplikacije. Umetnuti kod može biti kreiran i nekim drugim jezikom kojeg podržava korisnikov mrežni preglednik, primjerice HTML, VBScript, Java i Flash. Prostor za injektiranje zlonamjernog koda obično se nudi pri popunjavanju raznih polja za unos (formulara ili obrazaca) mrežne aplikacije i dolazi od strane korisnika aplikacije (Hrvatska akademska i istraživačka mreža, 2008).

Ovisno o načinu umetanja koda i oblika njegovog malicioznog djelovanja, XSS napade možemo podijeliti na: neustrajne (*engl. Non-persistent*) ili reflektirajuće (*engl. Reflecting XSS attack*), napade temeljene na DOM (*engl. Document Object Model*) modelu te na ustrajne (*engl. Persistent*) napade. **Neustrajni napadi** navode korisnika na posjećivanje posebno modeliranih hiperlinkova (*engl. Hyperlink*) u kojima je spremljen neki štetni programski kod. Ukoliko korisnik posjeti poveznicu u korisnikovom pregledniku, izvršit će se štetni kod. Najčešća namjera napadača pri ovoj vrsti napada je krađa sesijskih podataka (*engl. Session hijacking*), najčešće kolačića (*engl. Cookie hijacking*). Kako bi malicioznu poveznicu pokušali zakamuflirati, napadači mogu

zakodirati dio URL-a heksadecimalnim zapisom. Primjere možemo vidjeti u kodnom bloku 1 i 2.

```
http://testsite.test/<script>alert("TEST");</script>
```

Kodni blok 1. Primjer maliciozne poveznice koja u sebi sadrži *JavaScript* kod

```
http://testsite.test/%3Cscript%3Ealert%28%22TEST%22%29%3B%3C%2Fscript%3E
```

Kodni blok 2. Primjer prikrivene maliciozne poveznice

Ustrajni napadi imaju trajni utjecaj na korisnike aplikacije jer se pri ovakvoj vrsti napada maliciozni kod trajno ili do otkrića propusta pohranjuje u bazi podataka mrežne aplikacije. Tako se prilikom svakog učitavanja pohranjenog sadržaja pokreće umetnuti kod i pogađa veliki spektar korisnika. XSS DOM napadi funkcioniraju na način iskorištavanja propusta mrežnog preglednika koji omogućuje pohranjivanje i prikazivanje sadržaja mrežne stranice na lokalnom računalu korisnika. Ako je napadač uspio trajno umetnuti zlonamjeran kod u stranicu, tada može pristupati i prouzrokovati štetu na sustavnoj razini, te kompromitirati osobni sadržaj korisnika na lokalnom računalu.

Uzrok ove ranjivosti najčešće je propust razvojnih inženjera pri kodiranju stranice gdje se nije izvršila pravilna sterilizacija ulaznog sadržaja. Sterilizacija koda u ovom slučaju odnosila bi se na kodiranje posebnih meta znakova HTML jezika, primjerice znakova “<“,“ >“, “;“, i sličnih koji su rezervirani za generiranje HTML elemenata i konačnici izvršavanje klijentskog skriptnog koda. Sterilizacijom ulaznog sadržaja preglednik neće interpretirati meta znakove kao HTML sadržaj već kao običan tekst. S druge strane klijenti se mogu zaštititi od XSS napada onemogućavanjem izvođenja nekog programskog koda u pregledniku čime se sprječava izvođenje zlonamjernog koda, ali najčešće time gube djelomičnu ili potpunu funkcionalnost mrežne aplikacije. Također, korisnik bi trebao obratiti pozornost prije posjećivanja dugih, sadržajem sumnjivih, poveznica.

3.2.2. SQL injekcija

SQL injekcija (*engl. SQL injection*) napadačka je tehnika koja koristi sigurnosnu ranjivost kod pristupa mrežne aplikacije bazi podataka. Na taj način moguće je ugroziti sigurnost mrežne aplikacije koja konstruira SQL upite iz podataka unesenih od strane korisnika aplikacije (Hrvatska akademska i istraživačka mreža, 2008).

Ova ranjivost prisutna je kod dinamičnih mrežnih aplikacija koje poslužuju dinamički generiran sadržaj korištenjem SQL upita konstruiranih od unesenih podataka putem formi ili drugih kanala i usmjerena je isključivo prema aplikaciji, a ne prema korisnicima. Ukoliko mrežna aplikacija pravilno ne filtrira (sterilizira) dobivene podatke izbjegavanjem znakova posebne namjene u SQL sintaksi, napadač može konstruirati zlonamjerni SQL upit koji će kompromitirati bazu podataka jer se izvršava pod pravilima pristupa mrežne aplikacije. Najgori ishod ovakve vrste napada je ostvarivanje kompletne kontrole nad bazom podataka te kompromitiranjem i izmjenjivanjem njenog sadržaja. Ova vrsta napada ne mora nužno rezultirati prepoznatljivim malicioznim djelovanjem što napadaču omogućava prikriveno pozadinsko djelovanje. Primjer SQL naredbi na primjeru dohvaćanja korisnika iz baze podataka na temelju korisničkog imena i lozinke možemo vidjeti u kodnom bloku 3. U kodnom bloku 4 možemo vidjeti primjer implementiranog strukturiranog programskog koda. U kodnom bloku 5 vidimo primjer maliciozne injekcije na mjestu korisničkog imena koja završava otvorene navodnike, dodaje operaciju logičkog *ili* nad tvrdnjom koja je uvijek istinita te na posljetku završava naredbu i komentira ostatak pripremljenog koda. Kao rezultat, ova radnja će uvijek vratiti prvog korisnika iz tablice korisnika.

```
SELECT Ime, Prezime FROM korisnici WHERE ime = 'korisnik' AND
                    lozinka = 'lozinka';
```

Kodni blok 3. Primjer validne SQL naredbe

```
mysql_query("SELECT Ime, Prezime FROM korisnici WHERE ime
        ='".$ime." AND lozinka = '".$lozinka."');
```

Kodni blok 4. Primjer ranjivog PHP programskog koda u kojem se slijepo interpoliraju varijable imena i prezimena u SQL naredbu

```
SELECT Ime, Prezime FROM korisnici WHERE ime = '' OR 1=1; -- AND
                        lozinka = 'lozinka';
```

Kodni blok 5. Primjer injektirane sintakse - ' OR 1=1;

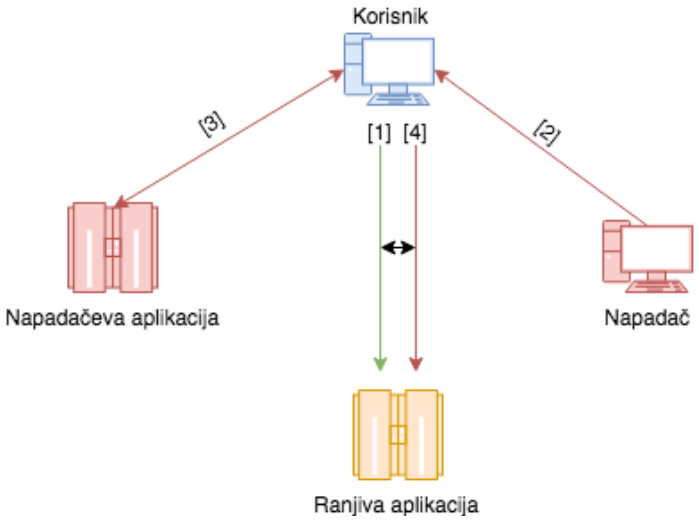
Prema načinu izvršavanja napada, SQL injekcije možemo podijeliti na slijepe i normalne. **Slijepa SQL injekcija** zasniva se na metodi pokušaja i pogrešaka na način da napadač nakon otkrivene ranjivosti pokušava otkriti podatke o infrastrukturi baze podataka (primjerice vrstu DBMS-a, verziju baze podataka ili relacijsku shemu baze) formiranjem posebnih SQL upita. Za razliku od **normalne SQL injekcije** gdje napadač dobiva povratnu informaciju u obliku strukturiranih podataka, u slijepim injekcijama napadač nema taj luksuz te pokušava kompromitirati sustav „na slijepo". Nepravilno strukturirani upiti vode ka prikazivanju povratnih informacija o pogrešci u sustavu, čime napadač može otkriti dio infrastrukturalnih informacija te prilagoditi SQL sintaksu koja se malo razlikuje između DBMS-ova.

Ovaj napad kao i XSS može se izbjeći vrlo lako i to sličnom metodologijom. Razvojni inženjeri trebali bi sterilizirati ulazne podatke kako rezervirani znakovi ne bi bili interpretirani kao dijelovi SQL upita. Također preporučuje se korištenje spremljenih procedura kojima se definira pripremljena struktura upita kod kojih se sama baza podataka bavi filtriranjem i sanacijom danih parametara. Nadalje, programeri bi trebali biti oprezni pri selekciji prikazivanja poruka o pogreškama u sustavu jer bi mogli kompromitirati aplikaciju. U konačnici, polja u bazi podataka koja su od važnog sadržaja (primjerice korisničke zaporke) trebala bi biti šifrirana kako bi se napadačima onemogućilo ili otežalo zlonamjerno iskorištavanje prikupljenih podataka iz baze.

3.2.3. CSRF ranjivost

CSRF (*engl. Cross-site request forgery*) oblik je ranjivosti koja kompromitira sve korisnike mrežne aplikacije. Napadač u ovom slučaju kreira malicioznu mrežnu aplikaciju te koristi klijentski program (preglednik) i ranjivu mrežnu stranicu kako bi izvršio neku radnju u ime druge osobe, u ovom slučaju korisnika mrežnog preglednika. Ranjivost je uglavnom prouzrokovana nesigurnim oblikovanjem komunikacije i ne provjeravanjem autentičnosti primljenih HTTP zahtjeva na poslužitelju. U ovom napadu napadač želi iskoristiti identitet osobe koja je autenticirana na nekoj mrežnoj aplikaciji kako bi izvršila zlonamjernu radnju. Ovaj napad srodan je XSS napadu, ali se ipak razlikuje u tome što se kod XSS-a iskorištava povjerenje koje korisnik ima u ranjivu stranicu, odnosno u sadržaj prikazan na njoj, dok se kod CSRF-a pak iskorištava povjerenje stranice u korisnika (Hrvatska akademska i istraživačka mreža, 2010).

Kao primjer, te ujedno ilustraciju koraka na slici 13, možemo navesti slučaj u kojem je korisnik autenticiran u bankovnoj aplikaciji (korak 1) koja omogućuje prijenos novca te je korisniku dodijelila kolačić u trajanju od jednog dana. Napadač posjeduje mrežnu aplikaciju u obliku foruma koja je programirana na maliciozan način tako da pri svakom kreiranju odgovora na raspravu s klijentskog pretraživača u pozadini radi HTTP zahtjev za transakcijom na bankovnoj mrežnoj aplikaciji (korak 3) dok istovremeno formira legitiman sadržaj na forumu. Korisnik na nagovor napadača ili spontano posjećuje mrežnu aplikaciju napadača i koristi njen sadržaj (korak 2). Korisnikov preglednik uputit će HTTP zahtjev pri svakom kreiranom odgovoru na mrežnom forumu te automatski poslati dodijeljeni kolačić (korak 4) koji se čuva na korisnikovom računalu. Ukoliko bankovna mrežna aplikacija ne provjerava podrijetlo zahtjeva, korisnik će biti autenticiran, a nenamjerna bankovna transakcija izvršena.



Slika 13. Generički primjer CSRF napada

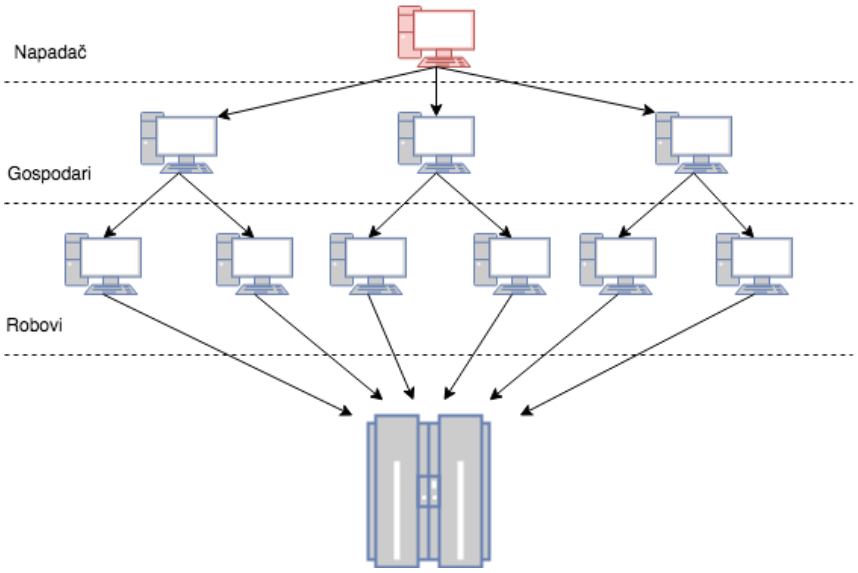
Kako bi se mrežne aplikacije zaštitile od ovakve vrste napada, potrebno je provjeriti i utvrditi autentičnost zahtjeva. Ukoliko želimo da korisnik obavlja neke radnje isključivo s izvorne domene aplikacije, prilikom popunjavanja i slanja formi s osjetljivim podacima potrebno je umetnuti CSRF značku (*engl. CSRF token*) odnosno poseban kod koji je dodijelila mrežna aplikacija, a kojim se potvrđuje da je zahtjev došao sa željenog izvora. Time će svaki zahtjev koji dolazi iz sumnjivog izvora i ne sadrži ili sadrži nevažeću CSRF značku biti odbijen. Također, pri radnjama od visokog rizika potrebno je izvršiti ponovnu autentikaciju i autorizaciju korisnika, primjerice elektronsko plaćanje. S druge strane, korisnik često ne može primijetiti CSRF napad, ali je formalno transparentan i može se utvrditi skeniranjem odlaznih zahtjeva s neke mrežne stranice. No, ovakve operacije nadilaze prosječno znanje korisnika aplikacije.

3.2.4. DoS napad

DoS (*engl. Denial of service*) napad ili napad uskraćivanja usluga je pokušaj napadača da učini nedostupnim računalo korisnicima kojima su namijenjene njegove usluge. Ovaj napad sastoji se od napora jedne ili više osoba kako bi trajno ili najčešće privremeno spriječili efikasno funkcioniranje mrežnog poslužitelja. DoS je implementiran

na način da se pokuša iskoristiti što više fizičkog resursa poslužitelja najčešće velikim brojem kompleksnih zahtjeva.

DoS napad najčešće se teško izvodi s jednog računala. Kako bi amplificirao svoj napad, napadač može koristi više pojedinačnih ili grupiranih računala koji će distribuirano opteretiti računalnu mrežu stvaranjem veće širine pojasa (*eng. Bandwith*) od žrtve napada. U tom slučaju govorimo u distribuiranom napadu uskraćivanja usluga ili DDoS (*engl. Distributed denial of service*). Da bi pokrenuo DDoS napad, napadač mora izgraditi mrežu računala koja će koristiti za stvaranje velikog prometa. To može ostvariti udruživanjem s drugim napadačima istih namjera ili otkrivanjem ranjivosti i preuzimanjem kontrole nad drugim poslužiteljima koje će koristiti kao alat napada. Preuzeta računala nazivaju se zombijima *(engl. Zombies*) ili robovima (*engl. Slaves*) jer su potpuno kontrolirana i čine radnje isključivo pod naredbom napadača odnosno gospodara (*engl. Masters*). Mreža takvih računala naziva se engleskim nazivom *Botnet* (Hrvatska akademska i istraživačka mreža, 2008). Shemu tipičnog DDoS napada možemo vidjeti na slici 14.



Slika 14. Primjer klasičnog DDoS napada

Također Hrvatska akademska i istraživačka mreža (2008) navodi nekoliko varijacija i metoda izvođenja DoS napada, a to su pod kodnim imenima: „Teardrop“, PDoS, DRDoS, „Nuke“, „Slowloris“, „Banana“, „Pulsing Zombie“ i „Smurf“.

Teardrop napad šalje oštećene IP fragmente s preklapanjem na ciljano računalo. U fragmente IP paketa upisuje se udaljenost od početka prvog paketa, što omogućuje ponovno sastavljanje paketa na drugoj strani. U ovom napadu napadač postavlja zbunjujuću udaljenost u jednom od fragmenata. Ako poslužitelj koji prima takav paket nema implementiranu kontrolu za takav slučaj, rezultat će biti pad sustava.

PDos (*engl. Permanent denial of service*) je vrsta napada koja se zasniva na pretjeranom iskorištavanju fizičkih resursa poslužitelja s ciljem oštećivanja ili uništavanja fizičkih komponenti udaljenog računala. Napad je najčešće usmjeren na usmjerivače, pisače ili drugo mrežno sklopovlje.

DRDoS (*engl. Distributed reflected denial of service*) generira istu količinu napada kao i klasičan DDoS napad, ali koristi efikasniju metodu. U ovom slučaju napadač šalje velikom broju poslužiteljima SYN pakete s lažiranom IP adresom odnosno IP adresom žrtve napada. Prema TCP protokolu poslužitelji su dužni uzvratiti SYN/ACK paketom. Ovim načinom zapravo se predstavljamo kao žrtva i reflektiramo napad koristeći druge poslužitelje (slika 15).

Nuke je stariji oblik DoS napada koji se bazira na slanju oštećenih TCP paketa prema računalu, čime se zahvaćeno računalo usporava te u konačnici dolazi do prekida rada. Ova vrsta napada bila je specifična za neke starije operacijske sustave s ranjivostima (Windows 95).

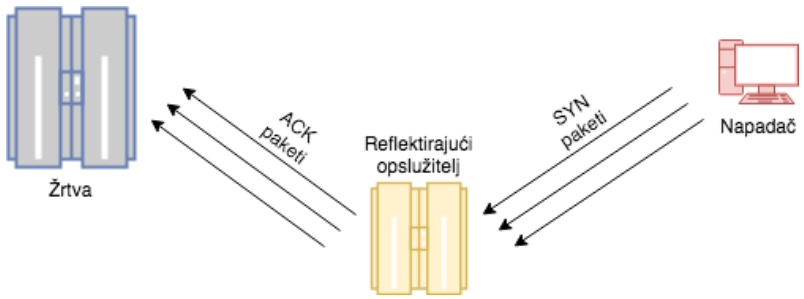
Banana je netipični napad koji uključuje preusmjeravanje odlaznih korisničkih zahtjeva natrag klijentu čime se klijentima uskraćuje usluga aplikacije jer zahtjevi nikad ne dolaze do poslužitelja. Ovaj napad najčešće iziskuje zauzimanje više čvorova s kojim graniči poslužitelj.

Pulsing zombie je izraz koji se odnosi na posebnu vrstu DoS napada kada je mreža podvrgnuta višestrukim *ping* porukama. Tako stanje rezultira degradiranom kvalitetom usluga i povećanim korištenjem resursa.

Smurf napad podvrsta je poplavljujućih (*engl. Flooding*) DoS napada na javni Internet. Napadač u ovom slučaju iskorištava loše konfigurirane mrežne uređaje primjerice one koje ne izvršavaju TCP rukovanje te lažiranjem izvorne IP adrese u IP paketima preplavljuje žrtvu koja prima odgovore s poslužitelja koje nije uputila.

Slowloris neobičan je napad koji se zasniva na spomenutoj perzistentnoj HTTP komunikaciji. Cilj napada je ostvariti što više paralelnih konekcija u kojima se HTTP zahtjev nikada ne pošalje do kraja. Svako zaglavlje trebalo bi završiti CR (*engl. Carriage return*) znakom odnosno praznom linijom. No, ukoliko se taj znak ne pošalje, poslužitelj očekuje nastavak zahtjeva i ostavlja vezu otvorenom u određenom periodu. Prije isteka tog vremena napadač ponovno šalje dio zahtjeva kojeg i dalje ne završava. Na taj način napadač pokušava zauzeti cijeli prostor iz bazena veza (*engl. Connection pool*) i onemogućiti drugim korisnicima da pristupe aplikaciji. Ovaj napad je suprotnost drugim napadima jer nije agresivan i invazivan s toga i nosi imena životinje ljenivca (*engl. Slow loris*) (Cloudflare, 2018).

Sigurna obrana od DoS napada ne postoji već se razvijaju inteligentni mehanizmi koji pokušavaju otkriti DoS napad u tijeku. Ovi alati najčešće su dio IDS-ova i vatrozida koji filtriraju ulazni promet i u moru zahtjeva pokušavaju prepoznati one maliciozne. Nekada takvi alati nisu dovoljni i potrebno je ljudsko nadziranje i brze reakcije u tekućim napadima. S druge strane, korisnici nenamjerno mogu postati dio DDoS napada bez njihovog saznanja jer su njihovu mrežu ili računalo zauzeli napadači. Također, ne postoji sigurna obrana od ovakvih slučajeva, no moguće je analizirati odlazni promet i utvrditi je li računalo korisnika „zombi“ računalo.



Slika 15. Reflektirajući DoS napad

3.2.5. Socijalni inženjering

Prema Hrvatskoj akademskoj i istraživačkoj mreži (2006), napade socijalnog inženjeringa obilježavaju postupci stjecanja informacija i podataka do kojih napadač

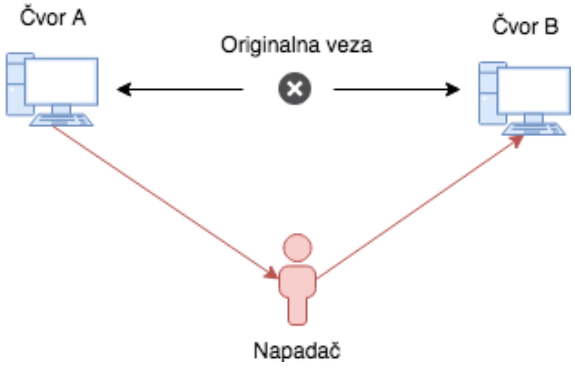
legitimnim putem ne bi mogao doći, pri čemu se ne iskorištavaju propusti računalnih sustava, već je napad usmjeren prema ljudskom faktoru. Filozofija socijalnog inženjeringa zasniva se na nekoliko metoda prijevare poput: uvjeravanja, lažnog predstavljanja, stvaranje stresnih situacija (u kojoj žrtva mora donositi brze i važne odluke), iskorištavanje moralne odgovornosti žrtve, iskorištavanjem želja za pomaganjem, te iskorištavanjem starih veza i kooperacija. Sve ove metode od napadača iziskuju potpuno drugačiji skup osobina i vještina od tradicionalnih računalnih napada. Napadač mora biti dobrog pamćenja, snalaziti se u razgovorima, biti karizmatičan ili čak šarmantan što mu daje prednosti prilikom izvođenja napada.

Najpoznatiji oblici izvršavanja ovog napada su: telefonski inženjering, socijalni inženjering korištenjem Interneta, pretraživanje otpada i forenzička analiza digitalnog otpada te uhođenje i špijuniranje. Putem telefonske linije napadaču je vrlo lako predstavljati se kao druga osoba i putem kvalitetno strukturiranih pitanja doći do informacija potrebnih za izvršavanje nekog drugog oblika napada. Automatizirana vrsta napada moguća je putem Interneta, gdje napadači najčešće koriste lažne poruke (*engl. Scam*) koje oglašavaju različite krivotvorene ponude i upozorenja. Time se žrtva navodi na unos osobnih podataka i zaporki ili pak otkrivanja nekih drugih relevantnih informacija. Ovakvi oblici napada nazivaju se engleskim nazivom *Phishing attack* koji u doslovnom prijevodu znači „pecanje“ što je odlična analogija za ono što napadač radi, baca udicu i očekuje da će čovjek napraviti pogrešku i zagristi.

3.2.6. MITM

MITM (*engl. Man in the middle*) napad je poznati oblik napada u kojem napadač presreće komunikaciju između dva čvora koja su uvjerena da direktno komuniciraju. U prošlosti ovaj napad bio je od velikog značaja jer protokoli i sigurnosni standardi nisu bili kvalitetno definirani. Jednom kad napadač postane „čovjek u sredini“ može izvoditi napade poput njuškanja (*engl. Sniffing*) mrežnog prometa, spomenute tipove DoS napada, otimanje sesije, trovanje DNS-a (*engl. DNS spoofing*) i slične.

Trovanje DNS-a proces je u kojem napadač podmeće lažnu IP adresu poslužitelja u odgovoru DNS servisa. Ovim napadom korisnik misli da pristupa validnoj domeni, ali zapravo komunicira s krivim računalom. Danas su ovakvi napadi većinom suzbijeni kriptografskim protokolima (TLS/SSL), a oni čvorovi s kojima nije uspostavljena šifrirana komunikacija izloženi su ovoj vrsti napada (Baloch, 2015). Shemu MITM modela napada prikazana je na slici 16.



Slika 16. Generički primjer MITM napada

3.2.7. Ostale ranjivosti

Postoji još niz napada koji iskorištavaju specifične propuste softvera i hardvera računalnog sustava. Takvi napadi najčešće žive i popularni su u određenom vremenskom periodu sve dok se ne kreiraju i ne rasprostrane sigurnosne zakrpe (*engl. Security patch*). Jedni od takvih ranjivosti su lažiranje IP adrese (*engl. IP spoofing*) i „Heartbleed".

Lažiranje IP adrese odnosi se na kreiranje IP paketa koji sadrže lažnu IP adresu s ciljem skrivanja identiteta pošiljatelja ili imitiranja nekog drugog računalnog sustava. Ova vrsta neučinkovita je kod HTTP i drugih zahtjeva koji iziskuju TCP vezu jer se nikada neće ostvariti uspostava veze zbog nepotpunog TCP rukovanja. S druge strane, ova metoda koristi se u spomenutom reflektirajućem DDoS napadu.

Heartbleed popularna je ranjivost TLS protokola otkrivena 2012. godine i otklonjena novijom verzijom istog protokola 2014. godine. S toga su svi računalni sustavi koji koriste stariju inačicu protokola i dalje otvoreni ovom napadu. Napad radi na vrlo niskoj razini implementacije protokola koji posjeduje funkcionalnost zvanu „Heartbeat" koja omogućava perzistentnu šifriranu TCP komunikaciju između dva računala.

Mehanizam iza funkcije nalaže da jedno računalo pošalje šifrirani podatak koje drugo računalo sprema u kratkoročnu memoriju te isti šifrirani podatak vraća prvom računalu čime potvrđuje da je komunikacija i dalje validna. Problem se krije u tome što se pri slanju zahtjeva šalje i informacija o duljini zakrivenog podataka koju drugo računalo koristi kako bi dinamički alocirao dio memorije potreban za pohranjivanje podataka. Pri slanju odgovora računalo prema istoj informaciji iz memorijskog međuspremnika (*engl. Buffer*) odnosno prostora u radnoj memoriji zaduženog za privremeni transfer podataka s jedne na drugu memorijsku lokaciju uzima definiranu duljinu i šalje prvom računalu. Tu varijablu napadač može iskoristiti kako bi lažirao (povećao) informaciju o duljini šifriranog podatka te time dobio dodatne informacije iz radne memorije drugog računala (OWASP, 2018).

Oblik sigurnosnog propusta gdje program tijekom pisanja ili čitanja podataka u/iz međuspremnika prelazi njegovu predviđenu duljinu (granicu) te time prepisuje ili iščitava memorijske adrese koje pripadaju nekom drugom programu ili drugom dijelu istog programa, naziva se preplavljivanje međuspremnika (*engl. Buffer overflow*) (OWASP, 2018).

4. Penetracijsko testiranje informacijskog sustava

Penetracijsko testiranje neizostavan je proces u životnom vijeku informacijskog sustava koji daje jasan pregled sigurnosti nekog informacijskog sustava. Ne samo da rezultat penetracijskog testiranja služi kao povjerljiva informacija o sigurnosnom stanju sustava, već u obliku certifikata može krajnjim korisnicima dati veću dozu sigurnosti i zadovoljstva pri korištenju sustava. Kroz sljedeća poglavlja upoznat ćemo se s osnovnim terminima vezanim uz penetracijsko testiranje, faze kroz koje svaki ispitivač, odnosno etički haker prolazi u procesu penetracijskog testiranja, industrijskih standarda i metodologija kojih se mora pridržavati te u konačnici nabrojati, opisati i usporediti neke od alata za automatsko i poluautomatsko penetracijsko testiranje.

4.1. Terminologija i podjela

Penetracijsko testiranje je tehnika procjene sigurnosti računalnog sustava ili mreže koja se temelji na oponašanju stvarnog napada. Prilikom testiranja, ovlašteni ispitivač provjerava metu, izvodeći različite vrste napada jednakim tehnikama koje bi koristio i da je stvarni napadač. Cilj mu je uočiti bilo kakvu ranjivost koju je moguće iskoristiti za ostvarenje neovlaštenog pristupa (Hrvatska akademska i istraživačka mreža, 2008).

Osobe koje se bave penetracijskim testiranjem najčešće su certificirani stručnjaci iz područja informatike i računarstva, točnije iz polja informacijske sigurnosti ili revizije informacijskih sustava. Prema Balochu (2015) svaka osoba koja se bavi otkrivanjem propusta i pogrešaka u računalnim i informacijskim sustavima naziva se *haker* (*engl. Hacker*), no ova titula ima dvije odnosno tri „strane medalje“. Ovisno o namjerama hakera i pridržavanja pravila hakerske etike, hakere možemo svrstati u tri skupine. Prvoj skupini hakera pripadaju bijeli šeširi (*engl. White hats*) ili etički hakeri (*engl. Ethical hackers*) i to su osobe koje svoje znanje koriste kako bi testirale i poboljšale računalne sustave te često surađuju s proizvođačima softvera. U našem slučaju to su osobe koje savjesno i uz odobrenje vlasnika vrše proces penetracijskog testiranja i u nastavku teksta ćemo im se referirati kao ispitivači. U drugu skupina hakera spadaju crni šeširi (*engl. Black hats*) ili

krekeri (*engl. Crackeri*) i to su zapravo hakeri koje namjerno uništavaju računalne i informacijske sustave s ciljem ostvarenja štete ili nekog oblika osobne dobiti. U treću skupinu hakera spadaju hakeri koji ne pripadaju ni u jednu ni u drugu grupu hakera i nazivaju se sivim šeširima (*engl. Grey hats*) i penetriraju računalne sustave iz zabave ili osobnog stručnog usavršavanja (bez dopusta).

Baloch također navodi i sekundarnu kategorizaciju hakera: skidove (*engl. ScriptKiddies*), elitne hakere (*engl. Elite hackers*), haktiviste (*engl. Hacktivist*) i već spomenute etičke hakere. Skidovi su hakeri kojima manjka elementarnog znanja o radu računalnih sustava te koriste gotove alate koje nisu u stanju reprogramirati, izmijeniti te prilagoditi za potrebe specifičnog slučaja koji nadilazi okvire napada obuhvaćenog nekim alatom. Elitni hakeri su komplementarni skidovima i posjeduju duboka znanja koja mogu iskoristiti za pisanje vlastitih alata i prilagođenih programa. Haktivisti se definiraju kao skupina hakera čiji motivi hakiranja proizlaze iz ekonomskih i političkih protesta, slobode govora, ljudskih prava i slično. Najpoznatija ovakva decentralizirana međunarodna skupina krije se pod nazivom *Anoynmous.*

Penetracijsko testiranje ne može dokazati da je sustav potpuno siguran, bez ikakvih mana. Ono što može je definirati sigurnosne propuste te utvrditi određenu granicu količine znanja i rada neophodnog napadaču za uspješan prodor u sustav. Uz poznavanje spomenutih informacija moguće je jamčiti sigurnost sustava samo u određenim situacijama, a nikako općenito i generalno. Bitno je i napomenuti da penetracijsko testiranje i tehnike koje se koriste pri penetracijskom testiranju variraju od sustava do sustava. Samim time spektar alata kojima ispitivač raspolaže aktivno se mijenja te su automatizirani alati ponekad beskorisni. S toga možemo reći da specifičnost i kompleksnost sustava zapravo korigira cijenu penetracijskog testiranja.

Prema Engebretsonu (2013) treba razlikovati dvije kategorije penetracijskog testiranja koje se razlikuju u načinu pristupanja testiranju te krajnjem cilju. U slobodnom prijevodu, tzv. testiranje bijele kutije (*engl. White box testing*) obuhvaća cjelokupno detaljno testiranje sustava gdje se ispitivaču omogućuju sva prava pristupa sustava, detaljan uvid u arhitekturu i izvorne kodove aplikacija, intervjue s razvojnim inženjerima i slično. Cilj ovakvoga testiranja je uočiti sve sigurnosne propuste sustava koji mogu dosezati i do

dubine do koje prosječni korisnik sustava ne može doprijeti. Glavni nedostatak ove metodologije je taj što traje poprilično dugo te ne kreira simulacije realističnih scenarija napada na sustav.

S druge strane imamo tzv. testiranje crne kutije (*engl. Black box testing*) kod kojeg ispitivač ne posjeduje nikakve informacije o sustavu prije početka testiranja. Ovakav način testiranja popularan je jer pristupa hakerskim napadima kroz realistične scenarije i time na najbolji mogući način testira razne defenzivne mehanizme za otkrivanje i suzbijanje napada, primjerice spomenuti vatrozidi, IDS-ovi i IPS-ovi računalnih mreža.

Kao i kod hakerskog entiteta, postoji sivo područje testiranja koje ne pripada niti jednom niti drugom ekstremu, a naziva se *testiranje sive kutije* (*engl. Grey box testing*). U tom načinu testiranja ispitivaču su ustupljene samo djelomične informacije o sustavu (Baloch, 2015).

Nadalje, Baloch dijeli penetracijske testove prema platformi i tehnologiji nad kojom se testiranje vrši. Tako poznajemo: testiranje računalne mreže, testiranje mrežnih aplikacija, testiranje mobilnih aplikacija, penetracijske testove društvenog inženjeringa te fizičke penetracijske testove. Penetracijski test društvenog inženjeringa ne spada u standardno testiranje samoga sustava, već je vektor napada usmjeren prema njegovim korisnicima korištenjem spomenutih *phishing* napada iz trećeg poglavlja. Fizički penetracijski test je nešto što se rijetko viđa u praksi, a podrazumijeva ostvarivanje fizičkog pristupa računalnim sustavima i mreži.

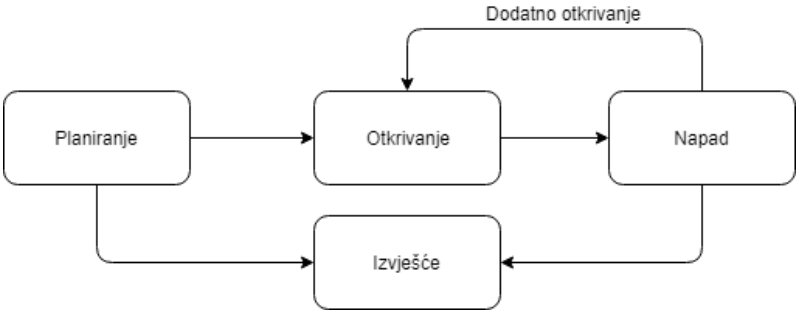
4.2.Standardi i metodologije penetracijskog testiranja

Iako postoje mnoge razlike u strukturi od sustava do sustava, te samim time razlike između penetracijskih testova na tržištu, postoje neki oblici normalizacije procesa penetracijskog testiranja. Iako se zbog specifičnosti penetracijskog testa teško slijepo pridržavati standarda, važno je držati se nekih osnovnih okvira. U svom priručniku o metodologiji penetracijskog testiranju Hrvatska akademska i istraživačka mreža (2008) navodi tri najkorištenije metodologije, dok Baloch (2015) spominje još jednu popularnu metodologiju mrežne zajednice OWASP.

OSSTMM (*engl. Open Source Security Testing Methodology Manual*) je priručnik koji detaljno opisuje proces penetracijskog testiranja. Cilj priručnika, koji su njegovi autori postavili još za vrijeme njegovog sastavljanja, je definiranje stroge metodologije penetracijskog testiranja, pri čemu se moraju zadovoljiti tri uvjeta: konzistencija, ponovljivost i pouzdanost rezultata. Ovaj priručnik podijeljen je u šest dijelova. Svaki dio predstavlja jedan fragment sigurnosti informacijskog sustava, a to su redom: informacijska sigurnost, sigurnost procesa, sigurnost internetskih tehnologija, sigurnost komunikacija, sigurnost bežičnih tehnologija te fizička sigurnost. Ako se ispitivač pridržava ovog standarda, isti mora analizirati i dokumentirati svih šest skupina odvojeno. Ako informacijski sustav ne sadrži neki od aspekata, tada u izvješću ta skupina nosi oznaku da se ne primjenjuje tj. "*NOT APPLICABLE*".

Nacionalni institut znanosti i tehnologije Sjedinjenih Američkih Država (*engl. National Institute of Science and Technology - NIST*) načinio je dokument s naslovom „Special Publication 800-42, Guideline on Network Security Testing". Cilj dokumenta je sistematično propisivanje elemenata testiranja sigurnosti u državnim organizacijama SAD-a. On identificira preduvjete koje je potrebno ispuniti za početak testiranja i preporučuje prioritete kojima je moguće testiranje provesti i s ograničenim resursima. Također, doprinosi izbjegavanju dvostrukog napora pružajući sustavan pristup cijeloj problematici. Dodatna mu je prednost i postojanje više razina testiranja koje, ovisno o organizaciji nad kojom se postupak primjenjuje, mogu dovesti do značajne uštede sredstava.

Prema Balochu NIST metodologija nalaže postojanje četiri faze: planiranje, otkrivanje, napadanje koje može završiti izvješćem ili odvesti u novo otkrivanje kao što možemo vidjeti na slici 17. Nadalje, faza napada dijeli se na dodatne faze: ostvarivanje pristupa, povećanja ovlasti, pretraživanje i popisivanje objekata sustava te instalacijom dodatnih alata koji bi kompromitirali sustav ili očuvali ponovni pristup istom. Detaljnije o fazama NIST metodologije govorit ćemo u narednom potpoglavlju.



Slika 17. NIST metodologija testiranja

ISSAF (*engl. Information Systems Security Assessment Framework*) je strukturirani radni okvir koji područje procjene sigurnosti računalnog sustava organizira u različite domene. Osim toga, on detaljno opisuje sasvim specifične testove koji se provode u svakoj od njih. Iako uključuje vrlo opsežan skup sigurnosnih procedura, još se uvijek smatra standardom u razvoju, te nije preporučljivo pouzdati se u rezultate njegovog provođenja.

OWASP (*engl. Open Web Application Security Project*) neprofitna je mrežna zajednica koja se bavi pisanjem dokumentacija i članaka, definiranjem metodologije te programiranjem alata i tehnologija korištenih u penetracijskom testiranju mrežnih aplikacija. OWASP priručnik za testiranje sadrži sve opise ranjivosti i testova koje bi trebalo provesti pri testiranju jedne mrežne aplikacije. Sav materijal ove zajednice je besplatan, a s konciznom dokumentacijom i metodologijom odlična je početna točka u karijeri penetracijskog testera mrežnih aplikacija.

4.3. Faze penetracijskog testiranja

Faze testiranja ovise o metodologiji koju prate i često se u literaturi razlikuju od autora do autora. No u suštini svaki od standarda obuhvaća sve radnje i korake kroz koje jedan ispitivač prolazi pri penetracijskom testiranju, samo su formalizirani na različite načine. Primjerice, Hrvatska akademska i istraživačka mreža (2008) opisuje osam faza testiranja koje odgovaraju NIST metodologiji, dok Engebretson (2013) definira vlastitu nestandardiziranu metodologiju u četiri faze. Za potrebe ovoga rada definirat ću i opisati

faze NIST metodologije jer pružaju solidan generički pregled procesa penetracijskog testiranja.

4.3.1. Prikupljanje informacija

Prikupljanje informacija *(engl. Reconnaissance*) je postupak koji uključuje korištenje Interneta za pronalazak informacija o ciljnom sustavu, organizaciji ili osobi, uz pomoć tehničkih (DNS/WHOIS protokola) i netehničkih metoda (pretraživači, novinske grupe, liste elektroničke pošte i sl.). Budući da se većina modernih informacijskih sustava nalazi u mrežnom okruženju, najčešće se do takvih informacija dolazi putem interneta koristeći javne izvore te kroz organizacije koje sadrže javne informacije (npr. porezne agencije, knjižnice i dr.). Što je više vremena utrošeno u prikupljanje informacija o meti, veće su šanse u postizanju rezultata u narednim fazama testiranja.

Engebretson (2013) navodi bazičnu podjelu faze prikupljanja informacija na aktivno (*engl. Active reconnaissance*) te pasivno prikupljanje informacija (*engl. Passive reconnaissance*). Aktivno prikupljanje uključivalo bi direktnu interakciju s metom, čime se povećava šansa za ranim otkrivanjem napada od strane obrambenih mehanizama sustava. S druge strane pasivno prikupljanje informacije obuhvaća prikupljanje informacija iz bilo kojih drugih već spomenutih izvora.

4.3.2. Mapiranje mreže

Mapiranje ili skeniranje mreže proces je u kojem pomoću različitih tehnika i prethodno prikupljenih informacija ispitivač pokušava rekreirati mrežnu topologiju sustava. To podrazumijeva otkrivanje podataka o svim aktivnim računalnim, odnosno poslužiteljima na kojima informacijski sustav postoji, određivanje razine propusnosti vatrozida, utvrđivanje operacijskih sustava računala, identifikacije svih ruta, domena i poddomena te pretraživanje vrata i servisa na mreži. U ovoj fazi ispitivač pokušava kreirati mapu cjelokupnog sustava, pronaći sve ulaze i izlaze sustava, odnosno definirati vektore hakerskog napada. Načini na koji će ispitivač najuspješnije dobiti uvid u žive poslužitelje

i servise sustava je skeniranjem vrata poslužitelja. Nakon što ispitivač utvrdi da je neki poslužitelj živ, slanjem posebnog ICMP (*engl. Internet control message protocol*) paketa te dobivenim povratnim rezultatom, skenirat će vrata poslužitelja skenerom vrata (*engl. Port scanner*). Ovi skeneri funkcioniraju na jednostavan način, a to je pokušaj uspostavljanja TCP veze (razmjene SYN/ACK paketa) na određenim vratima poslužitelja. Budući da smo u drugom poglavlju definirali korespondenciju vrata i servisa, lako je utvrditi koje servise određeni poslužitelj nudi.

4.3.3. Identificiranje ranjivosti

Nakon otkrivanja mrežne topologije ispitivač pokušava započeti penetraciju, odnosno pronaći točku ranjivosti iz koje će moći pristupiti sustavu. Aktivnosti kojima se postiže takva detekcija uključuju: identifikaciju ranjivih servisa korištenjem servisnih poruka (*engl. Banners*), pretraživanje ranjivosti s ciljem otkrivanja poznatih nedostataka[7], popisivanje otkrivenih ranjivosti, procjenu očekivanog utjecaja (klasifikacija pronađenih ranjivosti) i identifikaciju putova napada i scenarija za zloporabu.

4.3.4. Penetracija i dobivanje pristupa

Penetracija ili eksploatacija (*engl. Exploatation*) sustava je glavna faza penetracijskog testiranja u kojem ispitivač različitim metodama pokušava zaobići sigurnosna ograničenja i ostvariti pristup sustavu s bilo kakvim ovlastima. Proces penetracije podijeljen je u nekoliko faza: pronalazak programskog koda koji iskorištava ciljane ranjivosti, razvoj alata/skripti za testiranje, alata/koda za dokazivanje koncepta prilagodbi, alata/koda za dokazivanje koncepta potvrda ili pobijanje postojanja ranjivosti te dokumentiranje ranjivosti. Ovaj korak najčešće se zasniva na napadima grube sile (*engl. Brute force attack*) te napadima rječnicima (*engl. Dictionary attack*) kojima se pokušava doći do neovlaštenog pristupa nekom od servisa. Napadi grubom silom zasnivaju se na

[7] Informacije vezane uz poznate ranjivosti mogu se pronaći u proizvođačevim sigurnosnim oglasima ili u javnim bazama podataka.

generiranju i pokušaju autentikacije svim mogućim kombinacijama nekog niza znakova, primjerice svih malih i velikih slova duljine 10 znakova. Ovakva vrsta napada obuhvaća veliki broj pokušaja te time iziskuje velike kompjutacijske resurse kako bi se što više skratilo vrijeme potrebno za provedbu napada te je često nepraktično. S druge strane, napad neovlaštenog pristupa rječnikom, kao set konačnih vrijednosti, uzima unaprijed pripremljen rječnik koji može primjerice sadržavati popis svih riječi nekoga jezika ili lozinke koje su procurile u javnost kao rezultat nekog drugog (zlonamjernog) hakerskog napada. Primijetimo da mnogi od spomenutih servisa koji koriste FTP, POP, IMAP, SSH i slične protokole zahtijevaju neki oblik autentikacije u obliku korisničkog imena i lozinke čime su podložni ovakvim vrstama napada. U specifičnom slučaju mrežnih aplikacija, neovlašteni pristup prema servisima poslužitelja također se može ostvariti putem napada injekcija koda spomenutih u trećem poglavlju.

4.3.5. Povećanje ovlasti

Ako je ispitivač došao do ove faze, znači da je uspio zaobići sigurnosna ograničenja. U ovoj fazi želi se identificirati razina prava/ovlasti koju je ispitivač ostvario te analizirati i dokumentirati štetu koju može načiniti. Često se događa da je u prethodnim fazama dobiven pristup sustavu neke niže razine. U tom specifičnom slučaju, potrebno je izvesti mapiranje lokalnih ranjivosti (kao suprotnost ranjivostima baziranim na mreži), iskorištavanje ili razvoj dokaza koncepta (testiranog u izoliranom okruženju i primijenjenog na kompromitirani sustav). U ovoj je fazi cilj steći administratorske ovlasti temeljem prethodno stečenih ovlasti manje važnosti. Glavne prepreke tome cilju su primijenjene ispravke koje ispitivaču bitno umanjuju broj prisutnih ranjivosti, a time i mogućnost zlouporabe. Dodatnu prepreku predstavljaju alati za provjeru integriteta sustava (uključujući antivirusne alate), koji u nekim slučajevima mogu zaustaviti akcije ispitivača.

4.3.1. Daljnje popisivanje objekata

Daljnje popisivanje objekata (*engl. Enumerating further*) je proces prikupljanja dodatnih podataka o sustavu do kojih ispitivač nije mogao doći javnim pristupom u prvoj fazi. Podaci do kojih se najčešće dolazi u ovoj fazi jesu korisnička imena i lozinke korisnika, e-pošta liste sustava, podaci dobiveni prisluškivanjem i analizom prometa, prikupljanje korisnikovih kolačića (analizom prometa) te otkrivanje dodatnih ruta i direktorija sustava koji nisu dostupni javnosti.

4.3.2. Kompromitacija sustava

Jedna ranjivost u sustavu dovoljna je za izlaganje čitave mreže, neovisno o tome koliko je sigurna njena periferija. Svaki je sustav jak (u ovom slučaju siguran) onoliko koliko su jaki njegovi najslabiji dijelovi. U ovoj fazi ispitivač pokušava iskoristiti sve mogućnosti pristupa sustavu i kompromitirati ne samo sustav već i sve korisnike sustava ako je to moguće. Drugim riječima, ispitivač pokušava otkriti maksimalnu moguću štetu koju može napraviti.

4.3.3. Održavanje pristupa i skrivanje tragova

Održavanje pristupa i skrivanje tragova nezaobilazan su dio penetracijskog testiranja, a ispitivaču osiguravaju stalnu i trajnu prisutnost na kompromitiranom sustavu bez mogućnosti razotkrivanja. Ovaj korak uključuje primjenu sljedećih alata i tehnika: skriveni kanali, stražnja vrata (*engl. Backdoor*), *rootkit* alati, skrivanje datoteka, čišćenja zapisnika, poništenja provjere integriteta, poništenja antivirusnih alata.

Primjer stvaranja skrivenih kanala i stražnjih vrata je otvaranje novih netipičnih vrata (primjerice 12345) koji će biti znan samo ispitivaču. Uspostavom TCP veze s tim vratima ispitivač će u bilo kojem trenutku, do primjerice, spoznaje administratora sustava o otvorenim vratima, imati izravan pristup sustavu. *Rootkit* alati daju mogućnost skrivanja datoteka, pokretanja procesa i programa kao da nikad nisu bili instalirani na operacijski sustav poslužitelja. Zbog svoje izvanredne prirode skrivanja postojanja od strane

operacijskog sustava i samih korisnika operacijskog sustava, *rootkit* alati su u stanju vrlo lako zaobići nadzor i najsofisticiranijeg antivirusnog softvera.

4.4. Alati penetracijskog testiranja

Kako bismo si olakšali i ubrzali posao, zajednica računalnih stručnjaka i inženjera dizajnirali su niz gotovih komercijalnih i nekomercijalnih pomagala u svrhu olakšanja provođenja procesa penetracijskog testiranja. U tim pomagalima nalaze se čak kompletni operacijski sustavi koji nude niz alata za penetracijsko testiranje. Ti operacijski sustavi većinom su nekomercijalni. S druge strane postoje i razni aplikacijski paketi (programska okruženja) neovisni o operacijskom sustavu i platformi. Neki od najpopularnijih operacijskih sustava, aplikacijskih paketa i samostalnih alata prema *Software Testing Help* (2018) su: *Netsparker, Acunetix, Metasploit, Wireshark, w3af, Kali Linux, Nessus, Burpsuit, Cain & Abel, Sqlmap, John the ripper, Nmap, Hydra* i drugi . U narednom tekstu opisat ću najpoznatiji operacijski sustav *Kali Linux* i programsko okruženje *Metasploit* te navesti neke od prednosti i nedostataka ručnog i automatiziranog provođenja procesa penetracijskog testiranja.

4.4.1. *Kali Linux*

Kali Linux jedan je od najpoznatijih nekomercijalnih operacijskih sustava koji je razvijen s naglaskom na testiranje sigurnosne ranjivosti, penetracijskih testiranja i sigurnosne analize različitih mreža, a osnovala ga je, razvila i održala *Offensive Security* grupa. Kao što možemo zaključiti iz imena, bazirana je i jedna je od distribucija operacijskog sustava *Linux* temeljena na *Debian Linux* distribuciji.

Kali dolazi s više od 600 alata iz domene penetracijskog testiranja. Najčešće se koristi kao „live USB“ ili „live CD“ distribucija, odnosno ne instalira se direktno na tvrdi disk računala, već koristi po potrebi neovisno o postojećem instaliranom operacijskom sustavu, no to ne mora uvijek biti pravilo. Kategorije alata koje *Kali* posjeduje svrstavaju se u: prikupljanje informacija, procjenu ranjivosti, mrežne aplikacije, napad na lozinke,

alate za eksploataciju, detektiranje i lažiranje, održavanje pristupa, alate za izvještavanje te sustavne usluge. Osim što sadrži alate za penetracijsko testiranje, *Kali* sadrži i alate za napade na bežične mreže, obrnuto inženjerstvo, hakiranje *hardwarea* te digitalnu forenziku (Kali Linux, 2018).

4.4.2. *Metasploit*

Metasploit je projekt otvorenog koda, namijenjen analizi računalne sigurnosti. Osigurava informacije o sigurnosnim ranjivostima te pomaže u izvođenju penetracijskog testiranja i razvoju IDS potpisa. Njegov najpoznatiji potproizvod jest *Metasploit okruženje* (*engl. Metasploit framework*), razvojna platforma za izradu i izvođenje koda i alata za zloporabu na udaljenim računalima. Kao i drugi alati namijenjeni zaštiti informacijske sigurnosti, može se koristiti za zakonite ali i neovlaštene aktivnosti.

Ovo okruženja od 2003. Godine razvilo je nekoliko komercijalnih i nekomercijalnih inačica: *Metaspolit*, *Metaspoloit Pro*, *Metasploit Express* i *Metasploit Nexpose Ultimate*. Trenutna inačica nastala je potpunom izmjenom prethodnih i pisana je programskim jezikom *Ruby*. Razvio ju je *Metasploit LLC tim* odnosno kasnije *Rapid7* zajednica i dostupna je pod *BSD* licencom (Metasploit, 2018).

Metasploit okruženje sastoji se od alata, biblioteka, modula i korisničkog sučelja. Osnovna funkcija okruženja je učitavanje modula, što korisnicima omogućuje konfiguraciju i iskorištavanje modula za napad na ciljni sustav. U nomenklaturi *Metasploit* okruženja bitno je razlikovati pojmove modula*, exploit-a* i *payload-a*. Modul je dio *Metasploit* okruženja namijenjen iskorištavanju jedne specifične ranjivosti. Njemu pripada i određeni programski kod (*exploit)* posebno oblikovan iskorištavanju dotične ranjivosti. Cilj iskorištavanja ranjivosti je pokrenuti izvođenje posebno oblikovanog programskog koda na ciljnom računalu. Taj programski kod obično se naziva *payload,* univerzalan je za sve *exploit-e*, a specifičan za svaku računalnu arhitekturu. *Metasploit* okruženje sadržava nekoliko desetaka takvih kodova namijenjenih različitim funkcijama i oni se na ciljnom računalu koriste u kombinaciji s odgovarajućim ranjivostima (Hrvatska akademska i istraživačka mreža, 2008).

4.4.3. Ručno i automatizirano penetracijsko testiranje

Iako je donedavno proces penetracijskog testiranja iziskivao ispitivačevo visoko domensko znanje i najčešće provedbu ručnog testiranja, današnji ispitivači raspolažu nizom automatiziranih već spomenutih alata koji su lako prilagodljivi specifičnom slučaju bez posjedovanja iznimno visokog tehničkog znanja. Ručno testiranje, kao proces koji je sačinjen od niza zamornih i vremenski iscrpljujućih postupaka, zamijenjen je automatiziranim alatima, zahvaljujući dugogodišnjoj praksi i radu sigurnosnih stručnjaka. Automatizirano testiranje uvelike smanjuje količinu potrebnog vremena za provedbu jednog penetracijskog testiranja, no s druge strane ne može osigurati testiranje svih teško iskoristivih ranjivosti. U praksi najčešće se kombiniraju dvije navedene metode penetracijskog testiranja kako bi se osigurao optimalan omjer uloženog vremena i pokrivenosti testiranja računalnog sustava. Hrvatska akademska i istraživačka mreža (2008) u svome radu jednostavnom tablicom opisuje usporedbu ručnog i automatiziranog penetracijskog testiranja prikazanu u tablici 4.

	Ručno penetracijsko testiranje	**Automatizirano penetracijsko testiranje**
Proces testiranja	Radno-intenzivno, nedosljedno i sklono pogreškama, s nespecificiranim standardima kvalitete. Rezultati mogu značajno varirati od testa do testa. Općenito zahtijeva stručno sigurnosno osoblje za pokretanje testa i interpretaciju rezultata.	Brzo, jednostavno i sigurno. Eliminira pogreške zamornih ručnih zadataka. Centralizirano i standardizirano s ciljem dobivanja konzistentnih i ponovljivih rezultata.
Modifikacije mreže	Često se dogode mnoge izmjene na sustavu.	Sustavi se ne mijenjaju.
Iskorištavanje razvoj i upravljanje	Razvoj i održavanje baze zloporaba je vremenski skupo i zahtjeva značajnu stručnost. Javne zloporabe su sumnjive i nesigurne za pokretanje.	Proizvođač proizvoda (profesionalci) razvija, održava, nadograđuje i testira sve kodove te ih kontinuirano nadograđuje za postizanje maksimalne učinkovitosti.

	Ručno penetracijsko testiranje	**Automatizirano penetracijsko testiranje**
Čišćenje	Ispitivač mora zapamtiti i poništiti sve izmjene. Nakon napada mogu ostati otvorena stražnja vrata na sustavu.	Vodeći proizvodi nude iscrpno uklanjanje izmjena i stražnja vrata se nikad ne instaliraju.
Stjecanje i povećanje ovlasti	Zahtijeva se izmjena sustava, budući da se kod mora postaviti i prevesti na kompromitiranom računalu.	Korisnici mogu brzo prodrijeti dublje u mrežu. Kod se nikad ne mora postavljati na ciljno računalo i testovi se mogu provesti udaljeno.
Izvještavanje	Zahtijeva značajan trud, bilježenje i uspoređivanje svih rezultata ručno. Svi izvještaji moraju biti generirani ručno.	Iscrpni prikaz prethodnih događaja i pronađenih mana generiraju se automatski i prilagodljivi su.
Zapisivanje	Spor, težak, često netočan proces.	Automatski se snima detaljno izvješće o svim aktivnostima.
Obučavanje	Ispitivači moraju naučiti nestandardizirane, *ad-hoc* metode testiranja.	Korisnici mogu naučiti i instalirati alat za manje od jednog dana.

Tablica 4. Usporedba ručnog i automatiziranog penetracijskog testiranja

5. Studija slučaja

U ovome poglavlju prikazat ću praktični primjer i rezultate (primjer izvještaja) penetracijskog testiranja. Prema članku 266. kaznenog zakona „Tko neovlašteno pristupi računalnom sustavu ili računalnim podacima, kaznit će se kaznom zatvora do jedne godine“ (Hrvatski sabor, 2011), stoga mi je za korist ovoga rada informatička služba anonimne ustanove[8] ustupila ugovorom uređena prava na penetracijsko testiranje (studija) računalne mreže (slučaj), time i mrežnih aplikacija koje djeluju na aplikacijskom sloju. Sva saznanja dobivena ovim istraživanjem biti će predana ovlaštenoj informatičkoj službi u obliku izvještaja, a otkrivene ranjivosti sanirane prije javnog objavljivanja ovoga rada.

Cilj istraživanja je kvalitativnim metodama isključivo utvrditi, a ne iskoristiti ranjivosti računalnog sustava. Nadalje, svaki podatak koji bi kompromitirao identitet računalne mreže istraživanja u nastavku teksta biti će cenzuriran (ime domena, IP adrese poslužitelja, prepoznatljivi sadržaj mrežnih stranica, nazivi mrežnih direktorija i slično).

5.1. Metode i struktura istraživanja

Penetracijsko testiranje sustava biti će provedeno krnjom NIST metodologijom (fazama koje definiraju otkrivanje ali ne i iskorištavanje ranjivosti) definiranom u četvrtom poglavlju rada. Testu ću pristupiti bez prethodno prikupljenih ili ustupljenih informacija o mrežnom sustavu, s toga ovo testiranje spada u kategoriju *Black box* testiranja.

Provedba i rezultati testova biti će interpretirani s dva gledišta: pristupanjem mreže izvana i iznutra (korištenjem javne pristupne točke MN). Testiranje će biti provedeno korištenjem kombinacije automatiziranih alata i ručnih metoda penetracijskog testiranja. Za provedbu automatiziranih testova koristit ću *Metasploit* okruženje, dok ću neke testove sprovoditi ručnim metodama testiranja. Pri opisivanju testiranja u ovome radu pokušat ću ne izlaziti iz okvira ranjivosti definiranih u trećem poglavlju rada. Nastavak teksta bit će segmentiran prema odabranim fazama penetracijskog testiranja definiranom NIST

[8] Zbog mogućnosti kompromitacije sustava, ime ustanove mora ostati anonimno, s toga ću se u daljnjem tekstu na nju referirati kodnim imenom MN (meta napada).

metodologijom i zaključen poglavljem koje sadrži konačan izvještaj o rezultatima testiranja sustava.

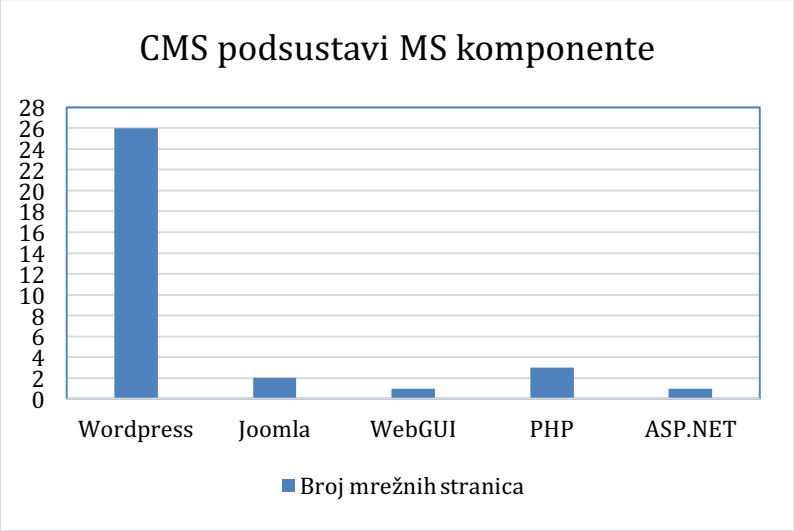
5.2. Prikupljanje informacija

Prvu fazu testiranja započet ćemo prikupljanjem informacija o meti iz javno dostupnih i transparentnih izvora. Za prikupljanje osnovnih informacija o računalnom sustavu koristit ću internetski preglednik *Google Chrome* te u njega ugrađenu ekstenziju *DevTools* namijenjenu ekstrakciji detaljnih podataka o prometu internetskog preglednika. Posjetom na glavnu mrežnu stranicu i bilo koju (pod)domenu ustanove koristeći spomenute alate možemo proučiti dolazna i odlazna zaglavlja HTTP poruka te utvrditi korišteni protokol, IP adresu poslužitelja, ulazna vrata poslužitelja, tip mrežnog poslužitelja, u nekim slučajevima verziju mrežnog poslužitelja, sadržaj zaprimljenih kolačića te površne podatke o programskoj potpori mrežnog poslužitelja (slika 18). Ove informacije poslužit će nam u izgradnji mrežne mape mete te stvoriti ulazne podatke potrebne za daljnje otkrivanje informacija o računalnom sustavu te sužavanju skupa vektora napada prema meti.

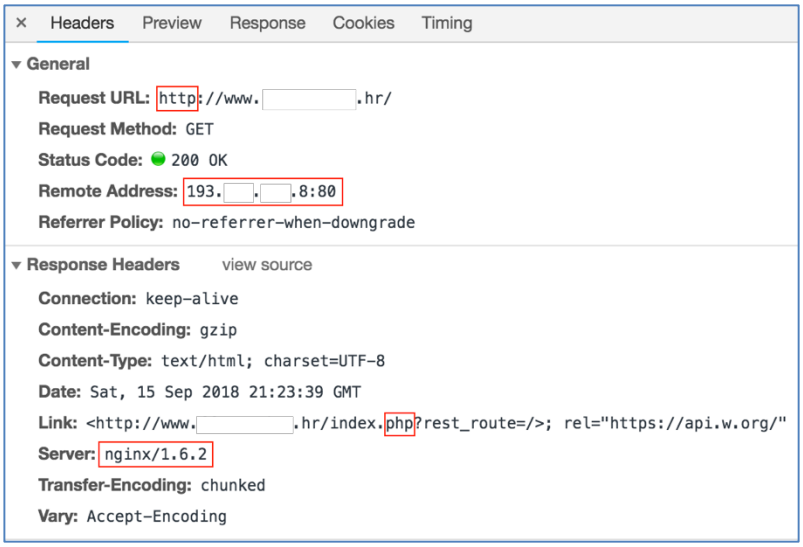
U našem slučaju pregledom ulazne mrežne stranice upoznao sam se s domenom MN koju podržava. Uočio sam te grafikonom 1 prikazao segregaciju mrežne stranice prema organima djelovanja na nekolicinu poddomena i direktorija koji upućuju na sustave (programskog okruženja) za upravljanje sadržajem (*engl. Content management system – CMS*). Jedan primjer ovakvog sustava omogućava funkciju kreiranja i modificiranja digitalnog sadržaja od strane urednika, dok s druge strane korisnicima omogućava napredno pretraživanje istog (Boiko, 2005). Uvidom u sadržaj HTML strukture stranica uspio sam utvrditi tipove i neke inačice CMS-ova očitavanjem meta oznake u zaglavlju HTML dokumenta (slika 19). Ručnim otkrivanjem sadržaja dolazim do saznanja o tri interaktivne aplikacije koje djeluju pod domenom MN. Pregledom HTTP zaglavlja i sadržaja odgovora mrežnog poslužitelja dolazim do primitivne strukture mrežnog sustava prikazanog u tablici 5. Svaka od komponenti sustava označena je kodnom oznakom koje ću koristiti kao alijas u narednom tekstu.

Komponenta	Mrežna stranica	Kataloška aplikacija	Email aplikacija	Edukacijska aplikacija
Kodna oznaka	MS	MA-1	MA-2	MA-3
Opis	Statička mrežna informativna stranica MN	Mrežna aplikacija namijenjena detaljnom pretraživanju informativnog digitalnog sadržaja MN	Mrežna aplikacija namijenjena korištenju usluge elektroničke pošte MN	Mrežna aplikacija namijenjena stvaranju i interakciji nad edukativnim sadržajem
IP adrese	193.xxx.xxx.7 193.xxx.xxx.80 193.xxx.xx3.8 193.xxx.xxx.34 193.xxx.xxx.28 193.xxx.xxx.82	193.xxx.xxx.5 193.xxx.xxx.226	193.xxx.xx2.8	193.xxx.xxx.91
Vrata	80/443	80/443	443	443
Protokol	HTTP/HTTPS	HTTP/HTTPS	HTTPS	HTTPS
CA certifikat	Validan	Validan	Validan	Validan
Mrežni poslužitelj	nginx	Apache	Apache	nginx
Verzija mrežnog poslužitelja	1.6.2	2.2.16	-	-
Operacijski sustav mrežnog poslužitelja	-	Debian	-	-
CMS	WordPress	WebGUI	-	Moodle
Verzija CMS-a	4.9.6	7.8.24	-	-
Programski jezik	PHP	Perl	PHP	PHP
Broj podsustava	32	1	0	0

Tablica 5. Osnovna struktura mrežnog sustava mete dobivena prikupljanjem informacija



Grafikon 1. Prikaz autonomnih CMS-ova unutar MS komponente



Slika 18. Primjer skeniranja HTTP zahtjeva MS komponente koristeći *Google Chrome DevTools* alat

```
<meta name="generator" content="WordPress 4.9.6" />
```

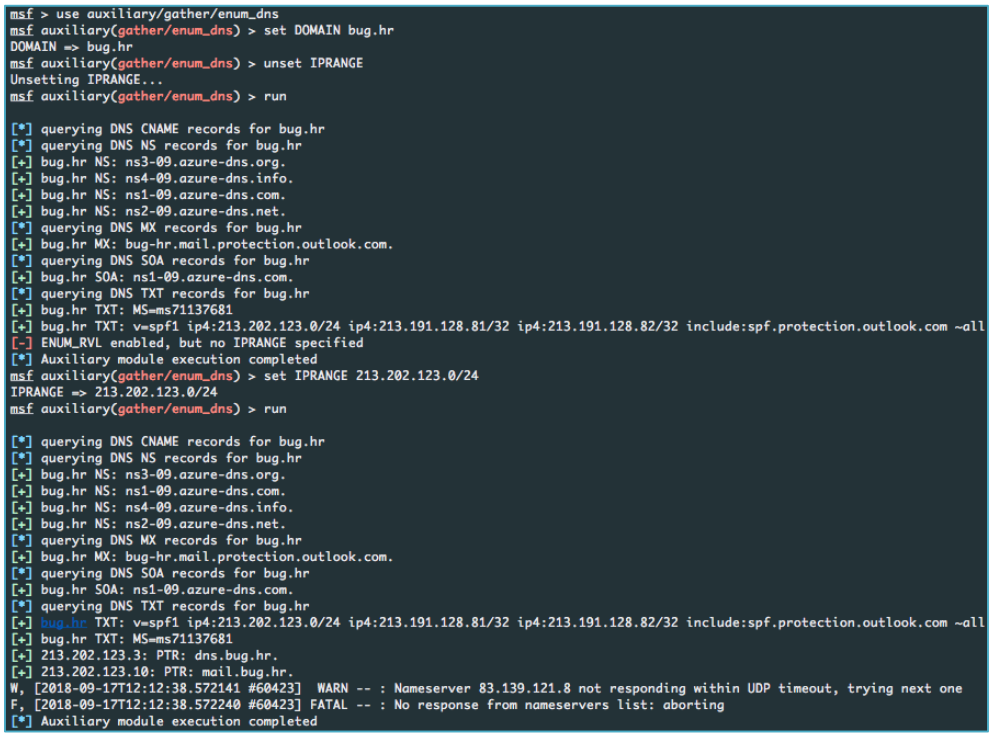
Slika 19. Primjer HTML *meta* elementa koji ukazuje na verziju *WordPress* CMS

5.3. Mapiranje mreže

U prijašnjoj fazi testiranja saznali smo grubu strukturu ciljanog mrežnog sustava na način da poznajemo adrese nekih od poslužitelja korištenih u infrastrukturi sustava. U ovoj fazi skeniranjem vrata i enumeracijom DNS zapisa utvrditi ćemo koje servise pojedini poslužitelj nudi mimo usluživanja mrežnog sadržaja korištenjem HTTP(S) protokola te koliko se ukupno poslužitelja nalazi u sustavu. Za potrebe ove zadaće koristit ćemo *Metasploit* modul pod nazivom *auxiliary tcp portscan* koji funkcionira na principu uspostavljanja TCP rukovanja s poslužiteljem nad skupom definiranih vrata. Parametri ulazne funkcije zahtijevaju listu IP adresa poslužitelja te listu ili raspon vrata za testiranje uz opcionalne argumente poput vremena razmaka između zahtijeva ili broj paraleliziranih procesa testiranja. Primjer definiranja postavki i naredbe izvršavanja nalaze se na slici 20. Također proučavanjem poddomena MS komponente i raspona IP adresa poslužitelja na kojima se nalaze, uočavam podudaranje prva tri segmenta IP adresa i enumeraciju zadnjeg segmenta što najvjerojatnije nalaže da se u sustavu možda nalazi više poddomena nego što je otkriveno prijašnjom fazom testiranja.

Za razotkrivanje i enumeraciju konačne liste pododmena sustava koristit ćemo *auxilary dns_enum* modul koji nudi funkcije DNS pretraživanja u oba smjera (primjer korištenja na slici 20). Razotkrivanje funkcionira na način da prvo dohvaćamo DNS zapise domene s DNS poslužitelja, a potom iskorištavamo popis IP adresa iz SPF (*engl. Sender Policy Framework*) datoteke zapisane unutar TXT DNS zapisa kako bi izvršili obrnuto DNS pretraživanje (*engl. Reverse DNS lookup*) putem IP adresa. SPF datoteka sadržava popis poslužitelja koji su autorizirani za slanje poruka elektroničke pošte u ime domene MN, čime vrlo vjerojatno dobivamo konačan popis svih poslužitelja MN.

Obrnutim DNS pretraživanjem dolazimo do potvrde hipoteze i brojke od četiristo trideset (430) poddomena sustava. Kasnijim mapiranjem mreže dolazimo do brojke od osamdeset i četiri (84) aktivna poslužitelja s kojima je otvorena komunikacija kroz barem jedan od skupa definiranih vrata. Radi ograničenog opsega rada, opis i prikaz daljnjih testova biti će usmjeren prema definiranim komponentama jer po subjektivnom mišljenju čine glavne okosnice računalne mreže mete.



Slika 20. Primjer obrnute DNS pretrage poddomena korištenjem *auxiliary enum_dns* modula nad bug.hr domenom

Pri prvom pokušaju mapiranja mreže, skeniranjem raspona vrata od jedan do deset tisuća (1-10000), nakon sto (100) pokušaja dolazi do odbijanja TCP(SYN) paketa od strane poslužitelja. Ova reakcija daje nam do znanja da je IDS poslužitelja prepoznao pokušaj mapiranja mreže grubom silom te blokirao IP adresu računala na mreži. Nakon toga računalo nije u mogućnosti uspostaviti nikakav oblik komunikacije s poslužiteljem, što iziskuje promjenu IP adrese i prilagodbu parametara testiranja.

S toga, nakon promjene IP adrese (spajanjem na drugu pristupnu točku) i definiranjem efektivne liste nama bitnih vrata (baze podataka, servisi elektroničke pošte, servisi za razmjenu podataka i slični) dolazimo do rezultata prikazanih u tablici 6. Možemo primijetiti da ne možemo pristupiti tradicionalnim brojevima vrata korištenih od strane baze podataka. To može značiti više stvari: mrežne aplikacije koriste eksterne poslužitelje na kojima su instalirane baze podataka, poslužitelj je dodijelio nekonvencijalni broj vrata bazi podataka ili, najvjerojatniji slučaj, gdje konfiguracija baze podataka ne dopušta pristup bazi podataka s niti jedne druge IP adrese osim lokalne 127.0.0.1 adrese poslužitelja. Što

znači da nikada nećemo moći ostvariti izravnu komunikaciju s bazom podataka osim ako se ne uspijmo infiltrirati u operacijski sustav poslužitelja kao korisnici ili otkrivanjem ranjivosti SQL injekcije na jednoj od aplikacija sustava. Pri provođenju istog postupka s pristupne točke MN, nije došlo do odbijanja TCP veza što nam govori da poslužitelji drugačije tretiraju vanjske i unutarnje zahtjeve.

```
msf > use auxiliary/scanner/portscan/tcp
msf auxiliary(scanner/portscan/tcp) > set RHOSTS 0.0.0.0
RHOSTS => 0.0.0.0
msf auxiliary(scanner/portscan/tcp) > set PORTS 1-5000
PORTS => 1-5000
msf auxiliary(scanner/portscan/tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   CONCURRENCY  10               yes       The number of concurrent ports to check per host
   DELAY        0                yes       The delay between connections, per thread, in milliseconds
   JITTER       0                yes       The delay jitter factor (maximum value by which to +/- DELAY) in milliseconds.
   PORTS        1-5000           yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS       0.0.0.0          yes       The target address range or CIDR identifier
   THREADS      1                yes       The number of concurrent threads
   TIMEOUT      1000             yes       The socket connect timeout in milliseconds

msf auxiliary(scanner/portscan/tcp) > run

[+] 0.0.0.0:              - 0.0.0.0:80 - TCP OPEN
[+] 0.0.0.0:              - 0.0.0.0:631 - TCP OPEN
[+] 0.0.0.0:              - 0.0.0.0:4000 - TCP OPEN
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Slika 21. Primjer skeniranja TCP porta korištenjem *auxiliary tcp portscan* modula

Komponenta	MS	MA-1	MA-2	MA-3
Vrata	21, 22, 53, 80, 443, 993, 995	22, 25, 80, 443	22, 25, 53, 80, 110, 443, 993, 995	25, 80, 443
Protokoli	FTP, SSH, DNS, HTTP, HTTPS, IMAPS, POP3S	SSH, SMTP, HTTP, HTTPS	SSH, SMTP, DNS, HTTP, POP, HTTPS, IMAPS, POP3S	SMTP, HTTP, HTTPS

Tablica 6. Prikaz rezultata skeniranja vrata mrežnih komponenti

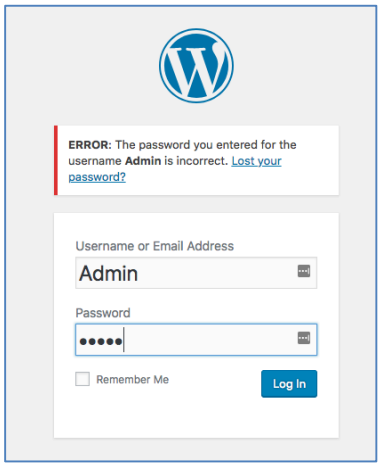
5.4. Identifikacija ranjivosti

Prikupljanjem informacija i mapiranjem mreže došli smo do vrijednih informacija na temelju kojih ćemo strukturirati plan razotkrivanja ranjivosti. Analizom komponenti utvrdili smo da je većina sadržaja popraćena CMS-ovima, koji sami po sebi ne ostavljaju

značajni koridor za ostvarivanje napada. Iza takvih sustava obično stoji jaka tehnička i razvojna potpora koja svakom novom inačicom programskog paketa pokušava otkloniti moguće sigurnosne propuste. Budući da su verzije većine CMS-ova ažurirane, nećemo trošiti vrijeme na pokušaje XSS i SQL injekcija nad formama tih sustava već ćemo takve ranjivosti pokušati otkriti ručnim postupcima nad sustavima neomeđenim CMS-om. CMS-ovima ću dati posebnu pažnju u kontekstu mehanizama otkrivanja korisničkih računa i mogućih ostvarivanja *Phishing* napada.

Slijedi pokušaj identificiranja forme aplikacija koje otkrivaju CSRF ranjivost analizom ulaznih polja HTML formi i HTTP zaglavlja kojima ćemo detektirati prisutnosti CSRF znački. Otkrivena vrata, odnosno servise, testirat ćemo otpornostima na napade prijave grubom silom za što ćemo koristiti module: *auxiliary ssh_login, auxiliary ftp_login, auxiliary http_login* te *auxiliary pop3_login*. U dogovoru s informatičkom službom, radi mogućeg onemogućavanja korištenje sustava izvršiti će se samo jedan oblik DoS napada, a to je *Slow Loris* napad nad komponentama MA-2 i MA-1 čiji poslužitelji koriste ranjive *Apache* mrežne poslužitelje. U konačnici analizirat ćemo SSL verzije, i koristeći *auxiliary open_ssl_heartbleed* modul pokušati razotkriti *heartbleed* ranjivost.

Proučavanjem mehanizma autentikacije WordPress CMS-a pronalazim kanal otkrivanja validnih korisničkih imena. Slika 22 ukazuje na postojanje korisničkog imena deskriptivnom povratnom porukom kojom nalaže da lozinka za uneseni korisnički račun nije točna. Sličan mehanizam razotkrivanja prikazan je koracima na slici 23, a bazira se na otkrivanju informacija o postojanju korisničkog računa povratnim porukama pri procesu promjene lozinke. Do validnih adresa elektroničke pošte korisnika CMS sustava došao sam prikupljanjem informacije iz prve faze, gdje sam vrlo lako dolazio do popisa adresa dostupnih na mrežnoj stranici poddomene (podsustava; organa MN). Iste adrese testirao sam prvotnim mehanizmom dok nisam došao do povratne poruka koja potvrđuje postojanje adrese u sustavu. Ova ranjivost omogućava ostvarivanje *Phishing* i drugih napada društvenim inženjeringom. Nadalje, gotovo svi podsustavi komponente MS ne koriste HTTPS protokol, čime su u kombinaciji s gore navedenim slučajevima otvoreni MITM napadima.



Slika 22. Primjer povratne poruke pri neuspješnoj prijavi sa validnim korisničkim imenom

Slika 23. Faze razotkrivanja validnog korisničkog imena ili email adrese korisnika WordPress CMS-a

Provođenjem napada prijave grubom silom uočene su razlike u pravima pristupa i kontrolnom tijeku suzbijanja napada pri izvođenju napad unutar i izvan mreže MN. Prema podacima iz tablice 7, možemo vidjeti da su napadi grubom silom omogućeni iz računalne mreže sustava. Konkretnije, radi se o napadu na FTP servis komponente MS i SSH servis

komponente MA-2. Pokušaji napada na ostale servise bili su neutralizirani u kratkom roku odbijanjem prava pristupa prijave.

Komponenta		MS	MA-1	MA-2	MA-3
SSH	UM	Ne	Ne	Da	-
	IM	Ne	Ne	Ne	-
FTP	UM	Da	-	-	-
	IM	Ne	-	-	-
POP3	UM	Ne	-	Ne	-
	IM	Ne	-	Ne	-
IMAP	UM	Ne	-	Ne	-
	IM	Ne	-	Ne	-
SMTP	UM	-	Ne	Ne	Ne
	IM	-	Ne	Ne	Ne

Tablica 8. Tablica istine napada prijave grubom silom nad servisima komponenata *(UM – unutar mreže, IM – izvan mreže)

```
msf > use auxiliary/scanner/ssh/ssh_login
msf auxiliary(scanner/ssh/ssh_login) > set RHOSTS 0.0.0.0
RHOSTS => 0.0.0.0
msf auxiliary(scanner/ssh/ssh_login) > set USERPASS_FILE /opt/metasploit-framework/embedded/framework/data/wordlists/default_userpass_for_services_unhash.txt
USERPASS_FILE => /opt/metasploit-framework/embedded/framework/data/wordlists/default_userpass_for_services_unhash.txt
msf auxiliary(scanner/ssh/ssh_login) > run

[-] Could not connect: The connection was refused by the remote host (0.0.0.0:22).
[!] No active DB -- Credential data will not be saved!
[-] Could not connect: The connection was refused by the remote host (0.0.0.0:22).
[-] Could not connect: The connection was refused by the remote host (0.0.0.0:22).
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Slika 24. Primjer neuspješnog napada prijave grubom silom nad SSH vratima (poslužitelj odbija zahtjev)

Komponente na čijim poslužiteljima djeluje Apache mrežni poslužitelj nisu pokazale ranjivost *Slowloris* napadom. Ova informacija ukazuje na ažuriranost ranjivog Apache mrežnog poslužitelja i urednu konfiguraciju mrežnog poslužitelja. Ključni faktori konfiguracije u obrani od ovakve vrste napada su: povećanje maksimalnog broja paralelno posluživanih klijenata, ograničenje veza koje dolaze s iste IP adrese, restrikcija minimalne brzine prijenosa podataka te maksimalno vrijeme održavanja veze. Napomena da je maksimalni broj postignutih paralelnih veza s poslužiteljem četiristo sedamdeset tri (473). Što nagovještava mogućnost izvođenja distribuiranog DoS *Slow Loris* napada koji bi mogao zauzeti cjelokupni bazen veza poslužitelja.

```
msf > use auxiliary/dos/http/slowloris
msf auxiliary(dos/http/slowloris) > set ssl false
ssl => false
msf auxiliary(dos/http/slowloris) > set rport 4000
rport => 4000
msf auxiliary(dos/http/slowloris) > set rhost 0.0.0.0
rhost => 0.0.0.0
msf auxiliary(dos/http/slowloris) > set delay 10
delay => 10
msf auxiliary(dos/http/slowloris) > run

[*] Starting server...
[*] Attacking 0.0.0.0 with 200 sockets
[*] Creating sockets...
[*] Sending keep-alive headers... Socket count: 200
[*] Sending keep-alive headers... Socket count: 200
[*] Sending keep-alive headers... Socket count: 200
[*] Sending keep-alive headers... Socket count: 200
[*] Sending keep-alive headers... Socket count: 200
```

Slika 25. Primjer DoS *Slowloris* napada korištenjem *auxilary slowloris* modula

Niti jedna od komponenti nije pokazala znakove curenja podataka pri slanju odnosno primanju *heartbeatova*, što govori o ažuriranosti *Open SSL* verzije poslužitelja.

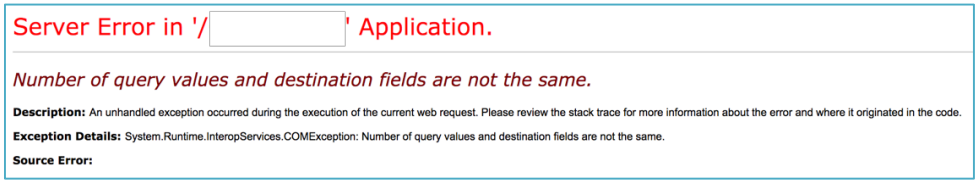
```
msf auxiliary(scanner/ssl/openssl_heartbleed) > set RHOSTS 31.13.84.40
RHOSTS => 31.13.84.40
msf auxiliary(scanner/ssl/openssl_heartbleed) > run

[*] 31.13.84.40:443       - Sending Client Hello...
[*] 31.13.84.40:443       - SSL record #1:
[*] 31.13.84.40:443       -     Type:    22
[*] 31.13.84.40:443       -     Version: 0x0301
[*] 31.13.84.40:443       -     Length:  81
[*] 31.13.84.40:443       -     Handshake #1:
[*] 31.13.84.40:443       -             Length: 77
[*] 31.13.84.40:443       -             Type:   Server Hello (2)
[*] 31.13.84.40:443       -             Server Hello Version:         0x0301
[*] 31.13.84.40:443       -             Server Hello random data:     16371d1a4e349bf0626f74c48a54ceca671827985a4b87f83d5d3da96917813b
[*] 31.13.84.40:443       -             Server Hello Session ID length: 32
[*] 31.13.84.40:443       -             Server Hello Session ID:      db835f70ee5cca3ccc108356b7b353176e82849bb06a2c6c4369e8f6e9af9619
[*] 31.13.84.40:443       - SSL record #2:
[*] 31.13.84.40:443       -     Type:    22
[*] 31.13.84.40:443       -     Version: 0x0301
[*] 31.13.84.40:443       -     Length:  2936
[*] 31.13.84.40:443       -     Handshake #1:
[*] 31.13.84.40:443       -             Length: 2932
[*] 31.13.84.40:443       -             Type:   Certificate Data (11)
[*] 31.13.84.40:443       -             Certificates length: 2929
[*] 31.13.84.40:443       -             Data length: 2932
[*] 31.13.84.40:443       -             Certificate #1:
[*] 31.13.84.40:443       -                     Certificate #1: Length: 1718
[*] 31.13.84.40:443       -                     Certificate #1: #<OpenSSL::X509::Certificate: subject=#<OpenSSL::X509::Name:0x007fb060cb6980>, issuer=#<OpenSSL::X509::Name:0x007fb06
0cb6958>, serial=#<OpenSSL::BN:0x007fb060cb6930>, not_before=2017-12-15 00:00:00 UTC, not_after=2019-03-22 12:00:00 UTC>
[*] 31.13.84.40:443       -             Certificate #2:
[*] 31.13.84.40:443       -                     Certificate #2: Length: 1205
[*] 31.13.84.40:443       -                     Certificate #2: #<OpenSSL::X509::Certificate: subject=#<OpenSSL::X509::Name:0x007fb060c84390>, issuer=#<OpenSSL::X509::Name:0x007fb06
0c84368>, serial=#<OpenSSL::BN:0x007fb060c84340>, not_before=2013-10-22 12:00:00 UTC, not_after=2028-10-22 12:00:00 UTC>
[*] 31.13.84.40:443       - SSL record #3:
[*] 31.13.84.40:443       -     Type:    22
[*] 31.13.84.40:443       -     Version: 0x0301
[*] 31.13.84.40:443       -     Length:  145
[*] 31.13.84.40:443       -     Handshake #1:
[*] 31.13.84.40:443       -             Length: 141
[*] 31.13.84.40:443       -             Type:   Server Key Exchange (12)
[*] 31.13.84.40:443       - SSL record #4:
[*] 31.13.84.40:443       -     Type:    22
[*] 31.13.84.40:443       -     Version: 0x0301
[*] 31.13.84.40:443       -     Length:  4
[*] 31.13.84.40:443       -     Handshake #1:
[*] 31.13.84.40:443       -             Length: 0
[*] 31.13.84.40:443       -             Type:   Server Hello Done (14)
[*] 31.13.84.40:443       - Sending Heartbeat...
[-] 31.13.84.40:443       - Unknown error
[-] 31.13.84.40:443       - Looks like there isn't leaked information...
[*] Scanned 1 of 1 hosts (100% complete)
```

Slika 26. Primjer neuspjele *heartbleed* eksploatacije nad facebook.com domenom

Ručnim ispitivanjem ranjivosti SQL injekcije nad svim komponentama i podsustavima MS komponente nailazimo na dvije (2) ranjive mrežne stranice (aplikacije)

unutar MS komponente. Otkrivanje ove ranjivosti provedeno je ručnim ispitivanjem ulaznih elemenata forme, odnosno unosom znakova koji bi mogli izazvati grešku (slika 27) parsiranja SQL naredbe (slijepa injekcija).



Slika 27. Primjer povratne informacije jedne od ranjivih mrežnih stranica komponente

Ručnim ispitivanjem XSS ranjivosti komponenti nalazim jednu mrežnu stranicu pod MS komponentom koja je podložna napadu. Međutim zbog karakteristika mrežne stranice nije izvodiva nikakva šteta nad sadržajem ili korisnicima sustava.

Ručnim ispitivanjem CSRF ranjivosti dolazim do saznanja da je autentikacijska stranica MA-2 komponente ranjiva pod CSRF napadom i napadom prijave grubom silom nad HTML formom.

5.5. Skraćeni izvještaj

Penetracijskim testiranjem MN u tri NIST faze (prikupljanja informacija, mapiranja mreže te otkrivanjem ranjivosti) dolazim do sljedećih relevantnih informacija. Skeniranjem mreže uočen je široki raspon od 430 (od kojih 84 aktivna) poslužitelja (virtualnih ili fizičkih računala) koji djeluju pod domenom MN. Proučavanjem poslovne domene računalni sustav MN dijelim u 4 rizične komponente.

Analizom svake od komponenti unutar računalne mreže MN lako dolazim do pregleda svih aktivnih vrata poslužitelja, aktivnih SSL certifikata i korištenih mrežnih protokola. Napadom prijave grubom silom nad dostupnim servisima poslužitelja dolazim do jednog FTP i jednog SSH servisa različitih komponenti.

Testiranjem komponenti na *Heartbleed* ranjivost SSL/TLS protokola dobivam negativne rezultate. Testiranjem dvije komponente (*Apache* mrežne poslužitelje) dobivam negativne rezultate na *Slowloris* DoS napad. Nadalje provjeravanjem CSRF znački i

ispitivanjem slanja forme putem HTTP protokola grubom silom nailazim na važnu CSRF ranjivost jedne od komponenti. Također nailazim na dvije mrežne stranice podložne napadu SQL injekcije. U analizi jedne od komponenti nailazim na niz (33) CMS podsustava (mrežnih stranica; poddomena). 78,8% navedenih podsustava koristi *WordPress* sustav za upravljanje sadržajem koji je u slučaju MN bio ranjiv na otkrivanje verzije i otkrivanje korisničkih računa podsustava.

6. Zaključak

U ovom iscrpnom radu čitatelja sam pokušao približiti tematici i problematici sigurnosti mrežnih aplikacija s različitih gledišta. Jasnim pregledom arhitekture računalnih mreža i mrežnih aplikacija generalno, stvorio sam podlogu za jasnim razumijevanjem nekih od sigurnosnih ranjivosti na nižim tehničkim slojevima. Definicijom i primjerima najučestalijih ranjivosti i napada nad mrežnim aplikacijama ukazao sam na relevantne i aktualne probleme sigurnosti mrežnih aplikacija. Nadalje, pored samog definiranja procesa i važnosti penetracijskog testiranja informacijskih sustava (mrežnih aplikacija) dokazujem ranjivost ciljanog mrežnog sustava MN. Smatram da su rezultati realizacije testiranja stvorili dovoljnu vrijednost ovome radu u svojoj svrsi. S druge strane otkrivanjem nekoliko sigurnosnih ranjivosti određenih komponenata ciljanog mrežnog sustava pomogao sam i osvijestio informatičku službu MN o mogućnosti i vrijednosti izvođenja profesionalnog penetracijskog testiranja sustava.

Nadam se će ovaj koncizan rad poslužiti budućim akademskim kolegicama i kolegama u potrebi jasnijeg razumijevanja ovog polja informacijske i računarske znanosti.

Literatura

1. Baloch, R., „Ethical hacking and penetration testing guide", CRC Press, 2015.
2. Bicsi, B., „Network Design Basics for Cabling Professionals", McGraw-Hill Professional, 2002.
3. Boiko, B., „Content Management Bible", John Wiley & Sons, 2005.
4. Cloudflare, „ Slowloris DDoS attack ", Pregledano 09.09.2018, https://www.cloudflare.com/learning/ddos/ddos-attack-tools/slowloris
5. Dujella, A., Maretić, M., „Kriptografija", Element, Zagreb, 2007.
6. Engebretson, P., „The basics of hacking and penetration testing", Elsevier Inc., 2013.
7. Fettinger, C., „ Multi-Factor Authentication: Explained ", Pregledano 24.08.2018, https://www.cyberstreams.com/multi-factor-authentication-explained
8. Hrvatska akademska i istraživačka mreža, „CSRF napadi", 2010.
9. Hrvatska akademska i istraživačka mreža, „DDoS napad", 2008.
10. Hrvatska akademska i istraživačka mreža, „Metodologija penetracijskog testiranja", 2008.
11. Hrvatska akademska i istraživačka mreža, „Socijalni inženjering", 2006.
12. Hrvatska akademska i istraživačka mreža, „Provjera XSS i SQL Injection ranjivosti Exploit Me skupom alata", 2008.
13. Hrvatski sabor, „Kazneni zakon", Narodne novine, Zagreb, 2011.
14. Kali Linux, „Official Kali Linux Documentation", Pregledano 11.09.2018, https://www.kali.org/kali-linux-documentation
15. Metasploit, „Getting started ", Pregledano 11.09.2018, https://metasploit.help.rapid7.com/docs
16. OWASP, „OWASP Top 10 – 2017: The ten most Critical Web Application Security Risks", Creative commons, Zagreb, 2017.
17. OWASP, „Heartbleed Bug", Pregledano 09.09.2018, https://www.owasp.org/index.php/ Heartbleed_Bug

18. OWASP, „Buffer overflow attack“, Pregledano 09.09.2018, https://www.owasp.org/index.php/Buffer_overflow_attack
19. Radovan, M., „Baza podataka : Relacijski pristup i SQL“, Informator, Zagreb, 1993.
20. Radovan, M., „Računalne mreže (1)“, Digital point tiskara d.o.o., Rijeka, 2010.
21. Radovan, M., „Računalne mreže (2) – Prijenos, mrežne usluge i zaštita“, Digital point tiskara d.o.o., Rijeka, 2011.
22. Tuđman, M. et. al, „Uvod u informacijske znanosti”, Informator, Zagreb, 1991.
23. Xplenty, „The SQL vs NoSQL difference: MySQL vs MongoDB“, Pregledano 19.08.2018, https://medium.com/xplenty-blog/the-sql-vs-nosql-difference-mysql-vs-mongodb-32c9980e67b2