Izrada CTF izazova za web aplikacije

Geci, Neven

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Neven Geci

Izrada CTF izazova za web aplikacije

ZAVRŠNI RAD

Varaždin, 2024.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Neven Geci

Matični broj: 0016143825

Studij: Informacijski sustavi

Izrada CTF izazova za web aplikacije

ZAVRŠNI RAD

Mentor:

Matija Kaniški, mag. inf.

Varaždin, rujan 2024.

Neven Geci

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U ovom završnom radu bit će opisan pojam web aplikacije, etičko hakiranje i najpopularniji operacijski sustavi za etičko hakiranje. Navest će se OWASP top 10 napada na web aplikacije i najčešći alati za otkrivanje web napada. Spomenut će se baza poznatih ranjivosti (*Exploit-DB*) i postupak prijave nove ranjivosti. Opisat će se CTF *(*engl. *Capture the Flag)* i kategorije CTF zadataka poput web, programiranje, forenzika, steganografija *Boot-to-root i Pwn.* Analizirat će se nekoliko postojećih CTF natjecanja (VulnHub, Hack The Box). U praktičnom dijelu rada izradit će se web aplikacija za CTF natjecanje. Natjecanje će se sastojati od više zadataka gdje će svaki zadatak zahtijevati veći nivo znanja. Detaljno će se opisati korištene tehnologije i programski jezici, zadaci na natjecanju, pravila natjecanja i sustav bodovanja zadataka.

Ključne riječi: CTF izazov; sustav za upravljanje CTF-om; web sigurnost; virtualizacija: PHP; Laravel; VirtualBox; Vagrant

Sadržaj

1. Uvod

CTF *(*engl. *Capture the flag)* izazovi ime je za skup natjecanja u kojima sudionici vježbaju i provjeravaju svoje znanje u raznim kategorijama informatičke sigurnosti. Zadaci unutar CTF natjecanja mogu biti u različitim kategorijama. Neke od kategorija su: Web aplikacije, kriptografija, forenzika, Pwn, *Boot to root* [1]. Nakon uspješnog rješenja zadatka natjecatelj dobiva zastavicu, odnosno skup znakova koji obično počinju s nekoliko znakova npr. „CTF“ te zatim u vitičastim zagradama sadrže varijabilan nasumični dio.

Ovaj završni rad će opisati CTF natjecanja i usporedit će metode i tehnike izvođenja natjecanja na primjeru dvije najpopularnije platforme za CTF natjecanja *Hack the Box* i *VulnHub*. Ranjivost je greška u programskom kodu sustava ili pak u procesu korištenja sustavom koja omogućuje njegovo iskorištavanje u zlonamjerne svrhe.

„Mrežni napad (engl. *Cyberattack*) je namjerni pokušaj krađe, otkrivanja, promijene, onemogućavanja ili uništenja podataka, programa ili drugih resursa koristeći se nedozvoljenim pristupom (engl. *Unauthorized access*) računalnoj mreži sustavu ili digitalnom uređaju“ [2]. U tim pothvatima koriste se alati za snimanje infrastrukture (engl. *Reconnaissance*) i izvođenje napada odnosno ranjivosti (engl. *Exploitation*). Takvi alati mogu biti javno dostupni kao npr. skup ranjivosti na platformi *ExploitDB*, dostupni unutar neke organizacije za privatnu uporabu ili dostupni za kupnju npr. na platformi *Zerodium*. Neki od najpoznatijih alata su *Nikto* i *sqlmap*.

OWASP (engl. *Open Web Application Security Project)* organizacija posvećena je edukaciji oko sigurnosti web aplikacija. U završnom radu se opisuju i kategoriziraju ranjivosti te predlaže moguća rješenja za iste.

Praktični dio ovog rada fokusirat će se na implementaciju sastava za upravljanje CTF izazovima te skupa CTF zadataka fokusiranog na kategoriju Web aplikacija. Izradit će se 5 zadataka koji imaju 5 zastavica. Prvi zadatak bit će pronalazak zastavice u opisu slike umetnute pomoću steganografije. Drugi će biti pronalazak zastavice nakon krađe sesije (engl. *Session hijacking)*. Treći će biti podizanje privilegija (engl*. Privilege escalation*) s pomoću ubacivanja SQL naredbi. Četvrti zadatak će podrazumijevati dobivanje pristupa ubacivanjem sustavskih naredbi (engl. *System commands*) dok će peti zadatak biti podizanje privilegija zbog neispravne konfiguracije.

CTF će se pokretati koristeći *VirtualBox* virtualni stroj, koji simulira dodatno računalo na lokalnoj mreži korisnika, tako mu daje pristup na virtualni stroj.

Razumijevanje sigurnosnih ranjivosti izuzetno je važno za programe u svrhu sprječavanja istih.

2. Web aplikacije

Izvor [3] definira web aplikaciju kao sustav koji se sastoji od dvije glavne komponente: strana klijenta (engl. *Frontend*) i strana poslužitelja (engl. *Backend*), odnosno klijentskog sloja i poslužiteljskog sloja [3]. Ta dva sloja komuniciraju putem aplikacijskog programskog sučelja (engl. *Application Programming Interface - API*). Grafička web sučelja se najčešće pišu s pomoću nekih od programskih okvira kao što su *Vue, React* i *Svelte*. Web aplikacije se povezuju na krajnje točke (engl. *Endpoint*) API-ja te im on šalje podatke u nekom od korištenih formata kao što su XML i JSON [3].

Također, može se primjenjivati i višeslojna arhitektura (engl. *Multi-layered architecture*) koja se služi API-evima drugih aplikacija kako bi proširila svoje mogućnosti. Višeslojna arhitektura omogućava lakšu integraciju s različitim sustavima, čime se postiže veća skalabilnost i fleksibilnost aplikacije.

2.1. Strana klijenta

Jednostranična aplikacija (engl. *Single Page Application – SPA*) je arhitektura web aplikacije koja učitava jednu HTML stranicu i dinamički ažurira tu stranicu dok korisnik obavlja interakciju s aplikacijom. SPA koristi AJAX (engl. *Asynchronous JavaScript and XML*) i HTML5 *(*engl. *Hypertext Markup Language)* za stvaranje fluidnih i responzivnih web aplikacija, bez potrebe za stalnim ponovnim učitavanja stranice. Ovaj pristup omogućava efikasnu interakciju s poslužiteljem dinamičkim mijenjanjem sadržaja potrebnog za korisnika. Time se povećava brzina izvršavanja aplikacija i poboljšava korisničko iskustvo. SPA arhitektura omogućava korisniku mnoge prednosti: smanjenje broja i veličine HTTP zahtjeva, programski kod cijele aplikacije učitan je kod prvog učitavanja aplikacije. Kasnijim interakcijama upravlja preglednik pomoću *JavaScripta* s klijentske strane. SPA arhitektura također omogućava visoki stupanj interakcije pomoću asinkronih *JavaScript* poziva (AJAX). SPA arhitektura standardizira načine upravljanja stanjem web aplikacije sa strane klijenta [4].

2.2. Strana poslužitelja

Strana poslužitelja web aplikacija je sloj za pristup podataka programske arhitekture i strukturira se u API pristupne točke. Postoji nekoliko široko rasprostranjenih paradigmi za komunikaciju s API-jevima. Trenutno najkorištenije paradigme za komuniciranje s API-jevima su: REST, *„GraphQL“* i poziv udaljenih procedura *(*engl. *Remote Procedure Call - RPC*) [3].

REST je najstariji i najrasprostranjeniji od tih paradigmi. REST se bazira na pozivu krajnjih točka s predefiniranim metodama kako bi se kreirao, pročitao, uredio ili obrisao neki resurs. Sljedeća slika ilustrira jednostavni REST API za upravljanje zadacima. Slika 1 prikazuje API za kreiranje, brisanje i dohvaćanje ili izmjenu specifičnog zadatka *(*engl. *Create, Read, Update, Delete – CRUD).* To se može vidjeti u stupcu zadatak. Pod stupcem *„metoda“* vidi se metoda koja se poziva. Pod stupcem *„putanja„* vidi se putanja na kojoj se nalaze akcije.

Zadatak	Metoda	Putanja
Kreiranje novog zadatka	POST	/tasks
Brisanje postojećeg zadatka	DELETE	/tasks/{id}
Dohvaćanje specifičnog zadatka	GET	/tasks/{id}
Pretraga zadataka	GET	/tasks
Ažuriranje postojećeg zadatka	PUT	/tasks/{id}

Slika 1: Primjer jednostavnog CRUD REST API-ja [5]

Zadatak (engl. *Task*) se kreira na putanji „/task“ uz metodu POST u zaglavlju zahtjeva.

Tijelo zahtjeva (engl. *Request body)* sadrži sve atribute zadatka u JSON ili XML formatu. Pomoću metode *„delete“* i parametra *„id* „briše se zadatak s određenim *id*-om. Tijelo zahtjeva može ostati prazno. Pomoću metode GET i parametra *„id“* dohvaća se jedan specifični zadatak, dok se pomoću metode GET bez parametara dohvaćaju svi zadaci. Pomoću metode PUT i parametra *„id“* uređuje se postojeći zadatak tako da se u tijelu zahtjeva proslijede svi atributi uređenog zadatka. Alternativno neki API-jevi implementiraju metodu PATCH koja očekuje *„id“* te samo one atribute koji se žele promijeniti. Atributi koji nisu navedeni ostaju isti kakvi su bili prije promjene.

GraphQL (engl. *Graph Query Language*) druga je paradigma za interakciju s API-jevima. Ima jednu pristupnu točku te u svojoj shemi ima stroge definicije tipova s njihovim atributima i vezama između tipova. *GraphQL* jezik koristi tipove u shemi kako bi znao koje upite može izvršiti. Ima slične osobine SQL-u, ima mogućnosti biranja poretka elementa, filtriranja te dohvaćanja povezanih elementa.

RPC najčešće se koristi u komunikaciji između dva računala, npr. u mikro-servisnim sustavima. Arhitektura definira funkcije sukladno korisničkim zahtjevima da budu javno dostupne s njihovom konvencijom poziva, odnosno argumentima. Zatim te javne funkcije može pozvati neki drugi sustav, neovisno o tome koji jezik ili operacijski sustav se koristi. RPC je „binarni“ protokol i kao posljedicu toga ima jako male pakete te je brz i efikasan. Najpoznatija implementacija RPC-a je *Googleov gRPC*.

3. Etičko hakiranje

Etičko hakiranje može se opisati kao „simulacija zlonamjernog napadača“ bez zlonamjernih namjerama. Kako bi se hakiranje moglo nazvati etičkim, napadač mora imati potrebne dozvole od strane organizacije čiju infrastrukturu napada. Uz dozvolu napadač obično dobiva i opseg testiranja. Opseg testiranja mora jasno definirati koje dijelove infrastrukture napadač smije testirati. Opseg također može definirati i dozvoljene ili pak zabranjene alate i tehnike tijekom testiranja. Jako je važno pridržavati se opsega testiranja te ostalih uputa i zakonskih okvira [6].

CTF-ovi imaju jako bitnu ulogu u etičkom hakiranju. Oni omogućavaju učenje i testiranje raznih alata i tehnika. Posebno se mogu izdvojiti „*Boot to root*“ CTF-ovi koji simuliraju ranjive sustave za napad. Napad se izvodi u potpunosti od snimanja infrastrukture mete (engl. *Reconnaissance*) do dobivanja potpunog pristupa nad sustavom.

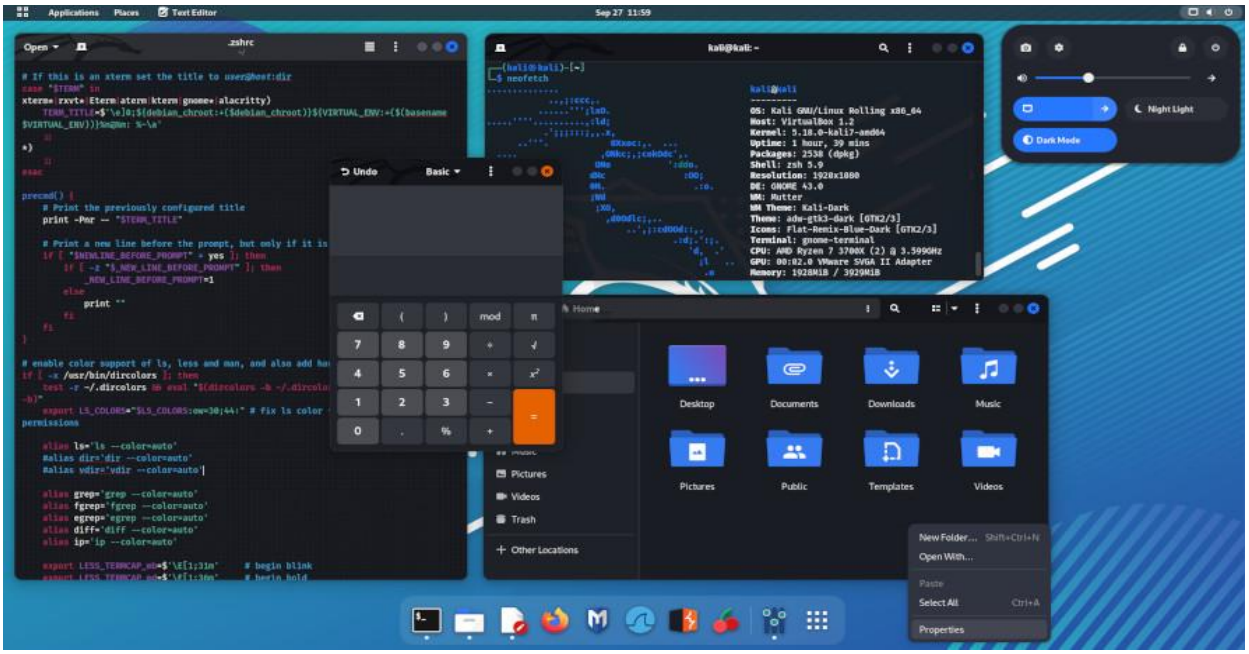
Postoje dva najčešća načina uključivanja etičkih hakera u profesionalno etičko hakiranje. Prvi način je da tvrtka ili fizička osoba sklopi ugovor s entitetom koji treba testiranje te provede testiranje unutar dogovorenog opsega. Etički haker o tom testiranju napiše izvještaj u kojem opisuje sve pronađene nesigurne ili potencijalno nesigurne dijelove sustava koji testira. Drugi način je sudjelovanje u programima za „lov“ na greške (engl. *Bug bounty programima*). Najpoznatije platforme za lov na greške su „*HackerOne*“ i „*Bugcrowd*“. To su platforme na kojima tvrtke objavljuju dozvoljene opsege napada za svoju infrastrukturu. Sudjelovati smiju svi, no jako je bitno da se pridržavaju dozvoljenog opsega i da se na odgovoran način prijavi pronađena greška. Nakon potvrde valjanosti pronađene greške od strane sigurnosnog tima organizacije koja je ranjiva, često slijedi novčana nagrada. Novčana nagrada ovisi o ozbiljnosti pronađene ranjivosti.

Odgovoran način prijave ranjivosti podrazumijeva da sigurnosni stručnjak kroz privatne kanale obavijesti oštećenu stranu. Dobra je praksa također da sigurnosni stručnjak dobro opiše ranjivost te pomogne kod njezine reprodukcije. Autor ranjivosti je dužan čekati da organizacija objavi zakrpu (engl. *Patch*) koja popravlja ranjivost te da oštećeni korisnici ažuriraju program na sigurnu verziju. Tek tada autor ranjivosti smije javno objaviti ranjivost. Neki autori odluče se za javnu objavu ranjivosti u slučaju da oštećena strana ne odgovara na privatne upite [7].

4. Operacijski sustavi za etičko hakiranje

Operacijskim sustavima za etičko hakiranje smatraju se sustavi koji imaju pred instalirane potrebne alate: programe, skripte i resurse kao što su liste riječi (engl. *Wordlist*) za napade grubom silom. (engl. *Brute force attack*). Dva najpoznatija operacijska sustava koja odgovaraju tom opisu su: „*Kali Linux*“ i „*Parrot OS*“.

4.1. Kali Linux



Slika 2: Kali Linux sučelje [8]

Kali Linux (prije BackTrack Linux) najpopularniji je sustav za etičko hakiranje po broju preuzimanja, podršci u vidu dokumentacije za učenje kao i podrške zajednice drugih korisnika. Sustav je otvorenog koda, baziran je na (engl. *Debian*) Linuxu i sadrži više od 600 alata za etičko hakiranje. Programi uključuju programe za napad na WIFI i Bluetooth mreže, npr. *Fluxion i Aircrack-ng*. Također sadržava puno alata za napad na web aplikacije kao što su [8]:

- *Nmap* - alat za skeniranje portova poslužitelj.
- *Nikto* - alat za skeniranje ranjivosti web aplikacija.
- *Sqlmap* služi za napad na SQL ranjivosti umetanja.
- *WPScan* - alat za skeniranje ranjivosti WordPress sustava za upravljanje sadržajem

- *Dirb* - alat koristi se za napad grubom silom (engl. *Brute force*) na web adrese poslužitelja ili virtualne poslužitelje.
- *BurpSuite* - alat za pregledavanje i manipulaciju HTTP zahtjeva.

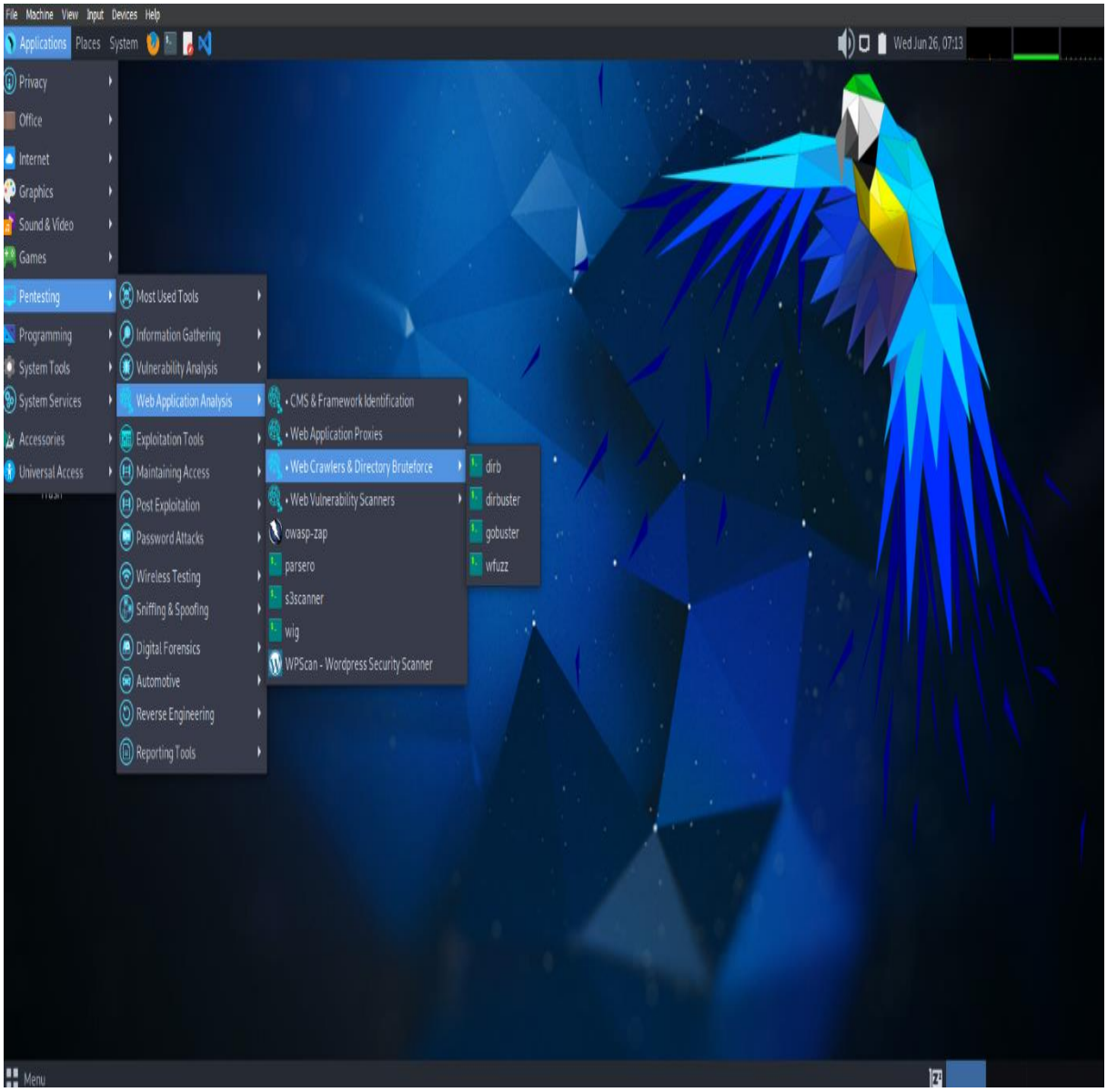
Kali Linux ima odličnu integraciju s postojećim alatima i poslužiteljima. Integriran je u Linux podsustav za *Windows*, *Amazon Web Services, Azure* i *Digital Ocean* računala u oblaku. Također ima službenu potporu za alate za virtualizaciju kao što su: *VirtualBox, VMware Workstation i Vagrant.*

Kali NetHunter je operacijski sustav za mobilne uređaje koji se fokusira na napade lokalnih računalnih mreža poput napada na WiFi i Bluetooth mreže, te napada zlonamjernog USB-a *(*engl. *Bad USB*). Napadi zlonamjernog USB-a su napadi kojima napadačev uređaj, u ovom slučaju mobilni uređaj, preuzima ulogu medija za unos, najčešće tipkovnice. Nakon toga u vrlo kratkom vremenu šalje korisni teret (engl. *Payload*) koji izvršavaju bilo koju akciju po želji napadača [8].

Kali Linux također ima odlične resurse za učenje u obliku dokumentacije i certifikata. Njihovi certifikati su vrlo cijenjeni i traženi kod zapošljavanja na pozicije povezane s etičkom sigurnošću. Certifikat OSCP (engl. *Offensive Security Certified Professional*) slovi kao jedan od najcjenjenijih i tehnički najzahtjevnijih certifikata u području web sigurnosti. Primjenjuju se primjeri stvarnim scenarijima [8].

4.2. Parrot OS

Parrot OS je operacijski sustav baziran na (engl. *Debian*) Linuxu. Ima nekoliko pod distribucija, no za etičko hakiranje koristi se „*Parrot Security*“ ili „*Hack the Box Parrot“* koji održavaju kreatori CTF platforme *Hack the Box*. *Parrot Security* pruža sličan skup alata za etičko hakiranje kao i *Kali Linux*, no uz to nudi i alate fokusirane na privatnost kao što je „*AnonSurf*“. Također pruža i alate za reverzno inženjerstvo (engl. *Reverse engineering*) kao što su „*Ghidra“* i „*GNUdebugger“*. Dodani su još i alati za digitalno forenziku poput „*binwaka*“ i „*autopsy-a*“. Također, valja napomenuti da *Parrot OS* zahtjeva manje hardverskih resursa od Kali Linuxa*. Parrot OS* može se preuzeti u obliku instalacijskog diska ili virtualnog stroja. Nema distribuciju pogodnu za mobilne uređaje [9].



Slika 3: Parrot OS sučelje (vlastita izrada)

Slika prikazuje Parrot OS s modernim, minimalističkim dizajnom koji kombinira tamne tonove i oštre kontraste. Plava pozadina sa stiliziranom papigom dodaje dinamičan vizualni element, dok je sučelje organizirano u pregledne izbornike, omogućujući brz pristup alatima za sigurnost i testiranje ranjivosti. Dizajn je funkcionalan i estetski ugodan, prilagođen tehničkim korisnicima

5. OWASP top 10 Web ranjivosti

OWASP je projekt namijenjen pružanju besplatnih resursa u svrhu edukacije o sigurnosti softvera npr. mobilnih aplikacije, oblaka, no primarno educira sigurnost web aplikacija. OWASP između ostaloga objavljuje i listu top 10 ranjivosti, a posljednja lista objavljenja je 2021. godine [10].

5.1. Nepravilna kontrola pristupa

Nepravilna kontrola pristupa *(*engl. *Authorization*) omogućava korisniku da bez „dovoljnih“ prava pristupa izvrši radnje. Postoji nekoliko glavnih razloga za takve napade. Resurs koji bi trebao biti dostupan samo određenoj skupini, dostupan je svima zbog nedostatka provjere autorizacije. Ako nekoj pristupnoj točki, na primjer upravljačkoj ploči, nedostaje provjera autorizacije, napadač može slobodno pristupiti tom resursu ako napiše točnu putanju, npr. „/admin-panel.php“. Uz nepravilnu provjeru pristupa napadač također može pogađati jedinstvene identifikatore resursa, primjerice identifikatori profila korisnika na društvenoj mreži, te ih mijenjati ili obrisati. Takav napad naziva se izravna referenca do objekta (engl. *Insecure direct object reference- IDOR*) [11].

S obzirom na način autentifikacije (provjere identiteta), moguća je i promjena trenutnih dozvola i prava, odnosno podizanje istih (engl. *Elevated access*). Ako su prava spremljena u kolačiće (engl. *Cookies*), moguće su bilo kakve izmjene istih. Ako se pak koriste sesije (engl. *Session*) kao način autorizacije, moguća je krađa sesije. Kad se koristi JWT *(*engl. *JSON Web Token),* moguće su promjene istog zbog nepravilnih konfiguracija ili validacija.

Neki od alata koji mogu pomoći pri ovoj kategoriji napada su „*Burp Suite*“ za manipulaciju HTTP zahtjevima ili pak „*JWT.io*“ za manipulaciju JSON web tokenima.

5.2. Kriptografski propusti

Napad Kriptografskih propusta (engl. *Cryptographic failures*) govori o nekoliko glavnih kriptografskih pogrešaka. Bitno je da se čitava komunikacija kriptira i da se ne šalje čisti tekst. Svi certifikati moraju biti validni i točno potpisani od strane nekog od izdavača certifikata. Certifikati su jako važni jer služe za potvrdu identiteta poslužitelja no u rijetkim slučajevima i klijenta [12].

Kriptografski algoritmi poput „MD5“ i „SHA1“ zastarjeli su i ne bi ih se trebalo koristiti jer su ranjivi na napade predviđanja (engl. *Padding oracle attack*) i napade grubom silom.

Kod programiranja je potrebno koristiti nasumične funkcije namijenjene za kriptografiju, te nisu podložne predviđanju izlaznih vrijednosti.

Neki od alata koji se mogu koristiti za napade na kriptografske propuste su „*Hashcat*“ i „*John the Ripper*“. Oni služe kako bi se nad sažetcima (engl. *Hash*) lozinki izveo napad grubom silom.

5.3. Ubacivanje zlonamjernog koda

Ubacivanje zlonamjernog koda tehnika je napada koja se događa zbog nedovoljne provjere korisničkih unosa. Napadač može u dokumentni objektni model (engl. *Document Object Model - DOM*) ubaciti *JavaScript* koji može ukrasti sesiju „žrtve“ ili pak poslati neki HTTP zahtjev u ime tog korisnika. Taj napad naziva se napad putem skripti (engl. *Cross-site scripting - XSS*) [13].

Napadač također može ubaciti i SQL programski kod i tako dobiti kontrolu nad upitima koji se izvršavaju u bazi podataka.

Također se može napasti i mehanizam za izradu podložaka (engl. *Templating engine*) te tako izravno pristupiti operacijskim sustavom kroz unos komandi operacijskog sustava (engl. *Command injection*).

Neki od alata koji se mogu koristiti za napade ubacivanja zlonamjernog koda su *SQLmap, XSSmap ili TPLmap.*

5.4. Nesiguran dizajn

Nesiguran dizajn (engl. *Insecure design*) opisuje kategoriju sigurnosnih ranjivosti u kojoj, čak i ako aplikacija nema ranjivosti u programskom kodu, ona je i dalje ranjiva. Na primjer, korištenjem sigurnosnih pitanja za potvrdu identiteta napadač može saznati odgovore na ta pitanja kroz javno dostupne podatke o osobi, ili pak tehnikama socijalnog inženjeringa [14].

5.5. Nesigurna konfiguracija

Napad nesigurne konfiguracije (engl. *Insecure configuration*) odnosi se na nesigurnu konfiguraciju okruženja u kojem se sustav izvršava. U ovu ranjivost može se svrstati jako puno raznih sustava. Nesigurna konfiguracija u oblaku podrazumijeva nesigurnu konfiguraciju sustava za upravljanjem identitetom i pravima pristupa. Do nesigurne konfiguracije može doći

i unutar okvira u kojem je aplikacija razvijena, npr. *ASP.net* ili *Spring Boot*, ili pak u operacijskom sustavu na kojem se aplikacija izvršava [15].

Neki od alata za testiranje nesigurne konfiguracije su: „*WPScan*“ za testiranje konfiguracije *WordPress*-a, „*LinPeaes*“ ili „*WinPeas*“ za testiranje konfiguracije Windows i *Linux* operacijskih sustava.

5.6. Ranjive i zastarjele komponente

Ranjive i zastarjele komponente (engl. *Vulnerable and Outdated Components*) u programima, kao i skripte za njihovo iskorištavanje, s vremenom postaju javni. Zbog toga je potrebno pravovremeno ažurirati sve komponente. Neki od poznatijih načina za pronalazak poznatih ranjivosti su „*MetaSploit*“ okvir ili „*ExploitDB*“ baza ranjivosti [16].

Korištenje zastarjelih komponenti može ozbiljno ugroziti sigurnost sustava jer napadači često koriste automatizirane alate za pretraživanje ranjivih verzija softvera. Prema OWASP-u, sigurnosni propusti u zastarjelim komponentama su među najčešćim uzrocima sigurnosnih incidenata. Stoga, osim redovitog ažuriranja, važno je implementirati i alate za kontinuiranu integraciju i isporuku (engl. *Continuous integration and continuous deployment- CI/CD*) koji mogu automatski provjeravati sigurnosne zakrpe i upozoravati na potrebne nadogradnje. Također, preporučuje se korištenje alata kao što su „*OWASP Dependency-Check*“ i „*Snyk*“ koji pomažu u identifikaciji ranjivosti u komponentama projekta.

5.7. Propusti autentifikacije i autorizacije

Ako aplikacija koja ima autentifikaciju dopušta automatizirane napade poput pronalaska elektroničkog identiteta (engl. *Credential stuffing*), napadač može iskoristiti popis postojećih korisničkih imena i lozinki kako bi pristupio korisničkim računima. Dopuštanje napada grubom silom može omogućiti napadaču da pogađa lozinke sve dok ne pronađe ispravnu. Korištenje nepromijenjenih, slabih ili poznatih lozinki, poput "Password1" ili "admin/admin", aplikaciju se izlaže riziku i čini istu podložnom tim napadima. Ako su postupci za oporavak lozinke preslabi, poput korištenja pitanja temeljenih na znanju, napadač može lako zaobići zaštitu i provjeru identiteta korisnika. Duljina lozinke i njezina kompleksnost bitno utječu na sigurnost lozinke. Preporučuje se korištenje upravitelja lozinki za njihovo generiranje i spremanje. Također je dobro da lozinka bude nasumična [17].

Pohrana lozinki u običnom tekstu ili slabo kriptiranih lozinki, predstavlja značajan sigurnosni rizik. Nedostatak više faktorske autentifikacije (engl. *Multi-factor authentication*) znači da napadač može pristupiti računu samo lozinkom, bez dodatne zaštite.

Izlaganje identifikatora resursa u URL-u (engl. *Uniform Resource Locator*) može omogućiti krađu sesije ako napadač dobije URL. Ponovno korištenje identifikatora resursa nakon prijave može dovesti do krađe sesije. Neispravno brisanje sesija ili autentifikacijskih tokena omogućava napadaču da koristi iste za neovlašteni pristup čak i nakon odjave ili neaktivnosti korisnika.

Neki od alata koji se mogu koristiti za izvođenje navedenih napada su „*Hydra*“ za web prijave i „*CrackMapExec*“ za Windows servise.

5.8. Propusti u integritetu softvera i podataka

Kako bi se spriječili propusti u integritetu softvera i podataka (engl. *Software and Data Integrity Failures*) potrebno je koristiti digitalne potpise ili slične mehanizme kako bi se provjerilo da softver ili podaci dolaze iz povjerljivog izvora i da nisu izmijenjeni. Također, treba osigurati da biblioteke i paketi, poput *npm-a* ili *Maven-a*, dolaze iz pouzdanih repozitorija. Ako postoji visoki rizik, preporučuje se korištenje internog provjerenog repozitorija [18].

Alati za sigurnost verzioniranja (engl. *Security versioning*) poput „*OWASP Dependency Check*“ ili „*OWASP CycloneDX*“, preporučuju se za korištenje kao provjera da komponente ne sadrže poznate ranjivosti. Važno je imati postupak provjere za promjene koda i konfiguracije kako bi se smanjila mogućnost uvođenja zlonamjernog koda ili konfiguracije.

CI/CD tijek rada mora imati pravilnu segregaciju, konfiguraciju i kontrolu pristupa kako bi se osigurao integritet koda tijekom razvoja i implementacije. Nepotpisani ili nekriptirani serijalizirani podaci ne smiju se slati nepouzdanim klijentima bez provjere integriteta ili digitalnog potpisa kako bi se spriječilo neovlašteno mijenjanje ili ponovno slanje tih podataka.

5.9. Propusti u sigurnosnim dnevničkim zapisima i monitoriranju

Propusti u sigurnosnim dnevničkim zapisima i monitoriranju (engl. *Security Logging and Monitoring Failures*) služe za pomoć pri detekciji, eskalaciji i odgovoru na aktivne sigurnosne napade. Bez bilježenja i nadzora napadi se ne mogu otkriti. Propusti u sigurnosnim dnevničkim zapisima i monitoriranju se javljaju kada nadzirani događaji (engl. *Audit events*) poput prijava, neuspjelih prijava i prioritetnih transakcija nisu zabilježeni. Upozorenja i greške uopće ne generiraju dnevničke zapise ili generiraju nedovoljne ili nejasne dnevničke zapise. Dnevnici aplikacija i API-ja nisu konfigurirani u svrhu pronalaska sumnjivih aktivnosti. Dnevnici se pohranjuju samo lokalno [19].

Razine izvještavanja za odgovarajuće upozorenje i procesi eskalacije do ljudske provjere nisu postavljeni ili nisu učinkoviti. Penetracijsko testiranje i skeniranje alatima za dinamičko testiranje sigurnosti aplikacija (engl. *Dynamic application security testing - DAST*) poput OWASP ZAP-a ne pokreću upozorenja. Posljedica toga je da aplikacija ne može detektirati i upozoriti na aktivne napade u stvarnom vremenu ili gotovo stvarnom vremenu.

5.10. Krivotvorenje zahtjeva na poslužiteljskoj strani

Krivotvorenje zahtjeva na poslužiteljskoj strani (engl. *Server-Side request forgery*) vrlo je opasna vrsta ranjivosti. Ono omogućuje napadaču komunikaciju s internim dijelovima infrastrukture kao što su baze podataka i servisi za predmemoriranje (engl. *Caching*). Ovakve ranjivosti mogu biti vrlo opasne i često dovde do izvršavanja koda na daljinu (engl. *Remote code execution - RCE*). Izvršavanje koda na daljinu je najvjerojatnije najopasnija vrsta ranjivosti [20].

Tijekom ovog napada napadač upotrebom krivo implementirane funkcionalnosti na web poslužitelju, može primjerice tražiti poslužitelja da šalje zahtjev računalima unutar lokalne mreže. Takva ranjivost dovodi do mogućnosti iskorištavanja računala koji nisu direktno dostupni na internetu.

6. ExploitDB

ExploitDB je baza isječaka naredbi za iskorištavanje ranjivosti, upita za *Google* (engl. *Google dorks*) kodova ljuske (engl. *Shellcode*) i istraživačkih radova (engl. *Research papers*). Isječci naredbi za iskorištavanje ranjivosti su skripte pisane u *Python-u*, *Ruby-u* ili nekom drugom programskom jeziku. Služe za automatizirano iskorištavanje poznatih ranjivosti. Kodovi ljuske su programski kodovi koji se koriste kod iskorištavanja ranjivosti u izvršnim datotekama. Oni su mali programi u *asembleru* koji imaju samo jednu namjenu, na primjer poslati ljusku sustava na specifični udaljeni poslužitelj. *Google dorkovi* su skup naprednih upita za pretraživanje *Google-a*, a u slučaju *ExploitDB-a* to su upiti za pronalazak ranjivih poslužitelja ili komponenata. Ranjivi poslužitelji mogu se naći prema zaglavljima poslužitelja ili pak posebno strukturiranim putanjama.

Predaja ranjivosti na *ExploitDB* ima nekoliko jasno definiranih pravila. Ranjivost treba poslati na e-mail: „submit@offsec.com“. Ona ne smije bit ciljana ni na jednu specifičnu web stranicu koja je javno dostupna. Jedna e-pošta smije sadržavati samo jedan prilog (engl. *Attachment*) u kojem se nalazi samo jedan isječak koda za iskorištavanje ranjivosti. Ne prihvaćaju se ranjivosti [21]:

- manipulacije dinamičkim bibliotekama (engl. *Dynamically linked library - DLL*)
- ranjivosti otkrivanja putanje (engl. *Path traversal*)
- ranjivosti otvorenih promjena putanje (engl. *Open redirect*)
- ranjivosti za koje je potreban najviši stupanj privilegija
- ranjivosti krađe klikova (engl. *Click-jacking*)
- ranjivosti bez identifikatora ranjivosti (engl. *Common vulnerabilities and exposures - CVE*).
- privremeni ili reflektivni XSS bez CVE-a.

E-pošta mora sadržavati: naslov ranjivosti, *Google dork*, datum otkrića (ako postoji), ime autora, poveznicu na web stranicu izvora programa, poveznicu za preuzimanje programa (ako postoji), verziju aplikacije u kojoj je pronađena ranjivost te CVE (ako postoji) [21].

7. CTF natjecanja

Capture the flag (CTF) su natjecanja u informatičkoj sigurnosti koja zahtijevaju rješavanje problema i razumijevanje iz različitih područja. Postoje različite vrste izazova u CTF natjecanjima: *kriptografija*, *steganografija*, *reverzno inženjerstvo*, *forenzika*, *Pwn* (iskorištavanje ranjivosti izvršnih datoteka sustava) i *Boot to root* (stjecanje potpunog pristupa sustavu). Ovi izazovi osmišljeni su kako bi testirali vještine sudionika te im omogućili učenje kroz praktično iskustvo [22].

CTF-ovi se mogu izvoditi u dva glavna oblika – *Jeopardy* i *Napad i obrana*. *Jeopardy* u slobodnom prijevodu znači '*opasnost*', a naziv potječe od popularne istoimene televizijske emisije zbog sličnog oblika. U kvizu "*Jeopardy",* natjecatelji biraju kategorije i pitanja različite težine i tako osvajaju bodove. Slično tome, u *Jeopardy* obliku CTF-ova timovi ili pojedinci biraju zadatke iz različitih kategorija kao što su: forenzika, reverzno inženjerstvo, kriptografija, mrežna sigurnost itd., i rješavanjem tih zadataka osvajaju bodove. Težina zadataka obično varira, a teži zadaci donose više bodova. Drugi oblik je *napad i obrana* (engl. *Attack and defense*) u kojem igrač nakon uspješnog rješavanja određenog zadatka predaje zastavicu. Zastavica je kratki tekst obično u prepoznatljivom formatu, npr. CTF{tajni-ključ}. Tekst tajni-ključ je nasumično odabran skup znakova koji je svaki put jednake duljine.

U obliku „*Napad i obrana*“ svaki tim dobije vlastitu kopiju ranjivih sustava koji sadrže identične izazove. Zastavica svakog tima za svaki izazov je jedinstvena. Ovaj oblik stvara zanimljivu dinamiku jer potiče timove da nakon pronalaska ranjivosti poprave istu na svom sustavu.

7.1. Web

Izazovi web kategorije najčešće se svode na iskorištavanje nekih od ranjivosti spomenutih u OWASP top 10 kako bi se ostvario veći nivo privilegija i samim time došlo do zastavice. Način iskorištavanja ovisi o ranjivostima pronađenim u fazi snimanja infrastrukture (engl. *Reconnaissance*), kao i jeziku, okviru i bibliotekama korištenim u kreiranju web aplikacije. Veliku ulogu igra i odabir tipa baze podataka, na primjer *SQL* ili *NoSQL* (relacijska, nerelacijska ili druga). Također, ranjivosti ovise i odabranom sustavu za baze podataka, primjerice *MySQL* ili P*ostgreSQL*. Uz bazu podataka web aplikacije mogu sadržavati i popratne servise, npr. sustave za međuspremanje (engl. *Caching*) kao što su *Memcached i Redis* [23]. Izazovi web kategorije obično se pokreću u *Docker* platformi na poslužitelju te igrač dobije pristup svojoj kopiji web stranice koju testira na sigurnosne ranjivosti.

7.2. Kriptografija

Kod kriptografskih izazova prvi, možda najbitniji korak, je prepoznati koji algoritam je korišten za šifriranje (enkripciju) ili enkodiranje poruke. Ponekad samim prepoznavanjem algoritma i korištenjem dekodera poput „*CyberChef-a*“ može se dobiti zastavica. Jedan od alata koji također može pomoći pri prepoznavanju algoritama je *FetherDuster* [24].

Isto tako postoje i napadi na enkripciju (engl. *Encryption attacks*). Napadač napadom grubom silom pokušava kriptirati puno potencijalnih lozinki istim algoritmom. Ako dobije enkripciju jednaku onoj koju želi dekriptirati, zna da je pronašao originalnu lozinku. Ako pak napadač ima pristup algoritmima enkripcije i dekripcije, kod nekih može reverznim inženjerstvom doći do ključa enkripcije. *XOR* (isključivo ILI) algoritam ranjiv je na ovaj napad. Posjedovanje dijela teksta koji piše u enkriptiranoj poruci također može pomoći kod zaobilaženja enkripcije. Na taj je način, zbog znanja kraja poruka njemačkih vojnika u Drugom svjetskom ratu, dešifrirana Enigma. Ranjivosti u algoritmima enkripcije mogu se pronaći i analizom izvornog koda (ako je isti dostupan) [25].

Igrač (natjecatelj) u ovakvim izazovima dobiva kriptiranu poruku. Cilj mu je prepoznati metodu enkripcije, zatim iskoristiti neku od ranjivosti koju ta enkripcija ima kako bi došao do prvobitne poruke. Na primjer, igrač može znati da su prva 4 znaka zastavice „CTF{„ dok su daljnji nasumični i završavaju s „}“ .

7.3. Steganografija

Steganografija je proces skrivanja poruke u naizgled običnim datotekama ili u meta-podacima datoteka. Steganografske poruke najčešće se skrivaju u slikama ili audio datotekama, tako se primjerice u *.svg* formatu može nalaziti u oznaci opisa slike. Skrivena poruka može se otkriti pomoću alata „*grep*“ ili „*exiftool*“, što znači da će igrač dobiti sliku, zvuk ili video. Iz dobivenog je cilj „izvući“ informaciju u formatu zastavice koja nije vidljiva na prvi pogled. Najčešće se koriste alati *ExifTool*, za provjeru metapodataka, ili uređivač heksadecimalnih vrijednosti za provjeru binarne datoteke u heksadecimalnom formatu [26] [27].

7.4.Reverzno inženjerstvo

Reverzno inženjerstvo proces je kojim se bez izvornog koda želi rekonstruirati program, kako bi se razumio način na koji funkcionira. Ono se najčešće provodi nad izvršnim datotekama. Za tu svrhu se koriste disasemblere, dekompajlere i debugere [28].

Disasembleri su programi koji pretvaraju izvršnu datoteku u format strojnog koda koji je čitljiv ljudima. To obično podrazumijeva pretvaranje instrukcija u njihove mnemonike, dok su adrese i brojevi zapisani u heksadecimalnom formatu. Jedan od najpoznatijih disasemblera je *IDA*.

Dekompajleri su programi koji strojni kod pretvaraju u pseudokod sličan C-u koji je još razumljiviji. Ako se iskoristi dekompajler na izvršnoj datoteci pisanoj u nekom programskom jeziku koji ima pravovremenu kompilaciju (engl. *Just in time compilation*) i izvršava se u virtualnom stroju, kao, dobit će se pseudokod sličan izvornom. Takvi programski jezici su na primjer C# ili Java. Najpoznatiji dekompajler je *Hex-rays decompiler*.

Debugeri su alati za analizu izvršnih datoteka tijekom njihovog izvršavanja. Dopuštaju postavljanje točaka prekida (engl. *Breakpoints*) te omogućuju uvid u vrijednosti stoga, hrpe i registara. Najpoznatiji debuger je *GNU Debugger* ili *GDB*.

Kod ovakvog tipa izazova igrač dobiva izvršnu datoteku i mora analizirati programski kod kako bi, na primjer, došao do valjanog ključa licence. Licence su najčešće u formatu kao kod starijih programa. Nekoliko nizova znakova jednake duljine odvojenih crticama. Cilj igrača je iz strojnog koda razumjeti algoritam provjere licence.

7.5. Forenzika

U izazovima forenzike analiziraju se podaci i meta-podaci kako bi se došlo do stanja i događaja u sustavu te kako bi se pronašla zastavica. Analizirati se može mrežni promet, slika diska, slika radne memorije, itd. U forenzici se koriste alati poput *Wireshark-a* za analizu mrežnog prometa, *Volatility-a* za analizu slika radne memorije ili pak virtualni strojevi za analizu slike diska [29].

Forenzički izazov funkcionira tako da se prikupe, primjerice, slike diska ili snimljeni mrežni promet od računala koji je bio pod simuliranim napadom. Kako se rekonstruiraju koraci napada, tako se otkrivaju i zastavice ili njihovi dijelovi. Na primjer, na putanji „C:\\AppData\Roaming\Microsoft\Word“ nalazi se međuspremnik za *Word* dokumente iz kojeg je moguća rekonstrukcija *.docx* dokumenta. Takve podatke nazivamo artefaktima (engl. *Artifacts*).

Iz navedenog primjera može se uočiti da forenzika zahtjeva odlično razumijevanje operacijskih sustava i njihovog načina rada, te kako odabrane akcije ostavljaju trajne zapise, npr. u dnevnicima.

7.6.Pwn

Pwn je kategorija koja podrazumijeva napade na izvršne datoteke. Kako bi se moglo efektivno napasti izvršnu datoteku, koriste se tehnike reverznog inženjerstva opisane u poglavlju 7.4. Nakon rekonstrukcije programskog koda reverznim inženjerstvom, u njemu se pronalaze ranjivosti. Neke od ranjivosti su preljev stoga, manipulacija hrpe ili pak manipulacija formatiranja znakovnih nizova [30]. Preljev stoga (engl. *Stack overflow*) je ranjivost pomoću koje napadač dobiva kontrolu nad pokazivačem stoga te samim time kontrolu nad instrukcijama koje će se izvršiti. Kod manipulacije hrpom može se manipulirati ili podacima na hrpi ili pokazivačima hrpe. Ako, se na primjer, sadržaj nekog od pokazivača promijeni na adresu funkcije, izvršit će instrukcije te funkcije. Ranjivost manipulacije znakovnim nizovima je posljedica slanja korisničkog unosa kao prvog parametra funkciji „*printf*“. Napadač u tom slučaju može iskoristiti modifikatore ispisa za čitanje i pisanje sadržaja u memoriji programa. Kada igrač igra ovakav tip izazova, uz njega obično dobije putanju do poslužitelja na kojem se nalazi izvršna datoteka, kopiju izvršne datoteke i ponekad kopiju „*libc*“ biblioteke koja je na udaljenom sustavu. Igrač dinamičkom analizom, odnosno korištenjem aplikacije u *GDB-u* ili pak reverznim inženjerstvom, mora pronaći i iskoristiti ranjivost, najprije na svom računalu, a zatim i na udaljenom poslužitelju.

7.7. Boot to root

Boot to root je kategorija koja postoji kako bi simulirala stvarne uvjete etičkog hakiranja. Cilj ove kategorije je postati administrator sustava i dobiti potpunu kontrolu nad istim. Za to je potrebno izvršiti snimanje infrastrukture (engl. *Reconnaissance*) i enumerirati svu javno dostupnu infrastrukturu. Zatim je potrebno pronaći inicijalnu ranjivost koja daje pristup dijelu interne infrastrukture i korištenjem novo otkrivene infrastrukture podići si privilegije. Eskalacija privilegija može se ponoviti nekoliko puta sve dok se ne dobije pristup administratoru. U *Boot to root* izazovu mogu se pojaviti svi ranije spomenuti izazovi i njihove kombinacije [31]. *Boot to root CTF* stroj će biti opisan u praktičnom dijelu ovoga rada.

8. Usporedba Hack the box i VulnHub CTF natjecanja

Dvije najpoznatije platforme za CTF-ove su *Hack the Box* (HTB) i *VulnHub* (engl. *Vulnerable By Design*). Zbog toga će njihova usporedba doprinijeti razumijevanju različitih načina održavanja CTF natjecanja.

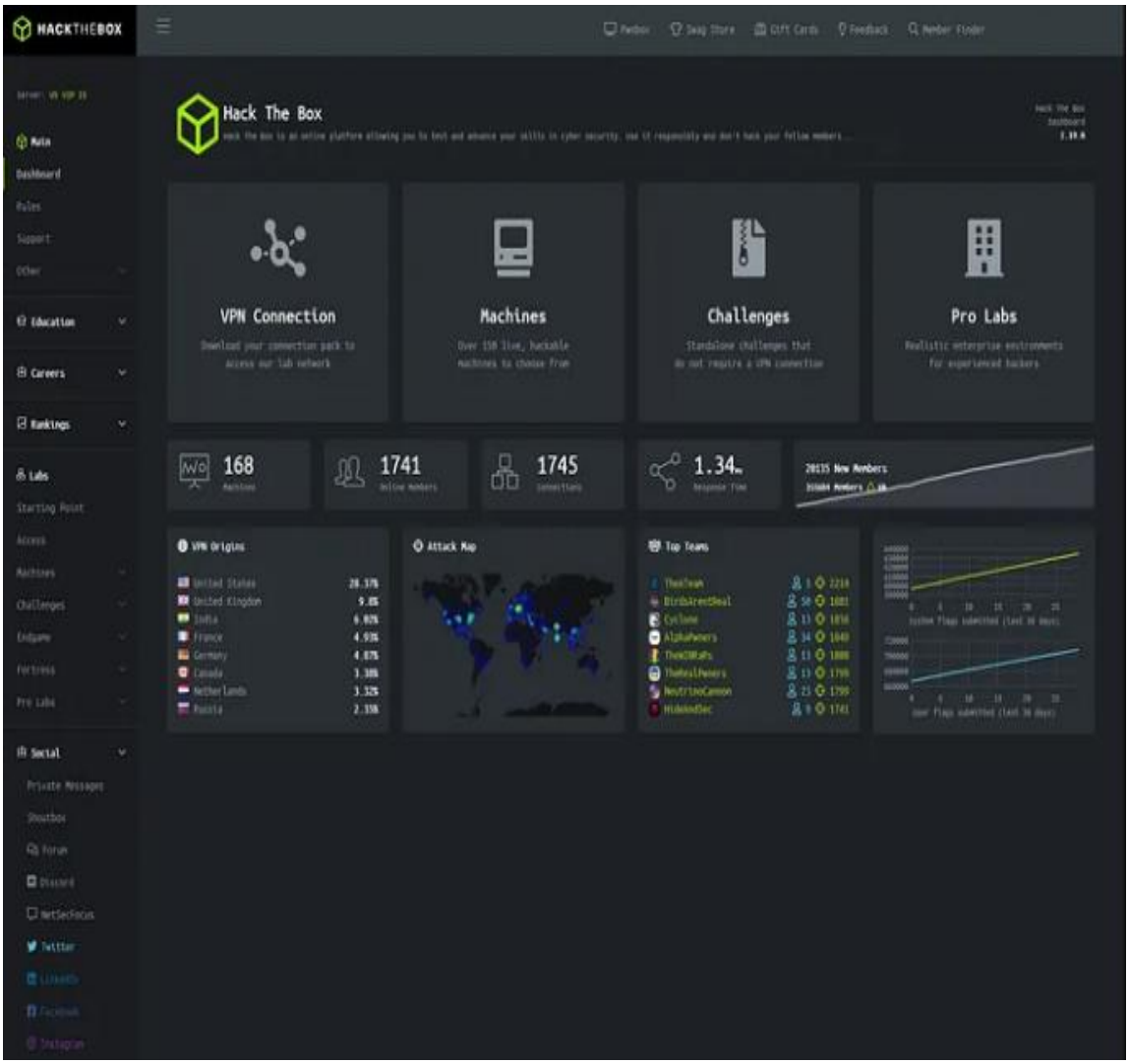
8.1. Hack the box

Hack the Box (HTB) je najpoznatija platforma za održavanje CTF natjecanja. Njihova izrazito poznata kategorija je „*Boot to root*“ koja sadrži slike virtualnih strojeva koji se mogu pohvaliti visokom sličnošću sa stvarnim slučajevima. Svaka slika virtualnog stroja sadrži dvije zastavice. Zastavicu nisko privilegiranog korisnika i zastavicu administratora ili *root-a*. Slike strojeva variraju po težini, operacijskom sustavu te programima i tehnikama koje je potrebno razumjeti da bi se iskoristile njihove ranjivosti. Težine idu redom: lako, srednje, teško i suludo teško. Neke od tehnologija koje se mogu pronaći na slikama strojeva su različite vrste web poslužitelja pisane različitim programskim jezicima, aktivni direktorij, protokoli (npr. SFTP, SMB, DNS) i programi kao što su *Grafana*, *Jenkins Nagios*, i sl. Također se mogu naći sustavi za upravljanje sadržajem poput *WordPress-a* i *Joomla-e*. Svaka slika stroja raspoloživa je u ograničenom vremenu. Nakon završetka vremena briše se i prestaje biti dostupna za besplatno korištenje te prestaje donositi bodove natjecateljima. HTB naglašava natjecateljsku komponentu svojih CTF-ova te je visoka pozicija natjecatelja na njihovim ljestvicama značajno postignuće [32].

Platforma uz „*Boot to root*“ pruža i CTF-ove forenzike. Takve CTF-ove nazivaju *Sherlock CTF-ovi*. Naziv je takav jer igrači primjenom forenzičkih metoda moraju otkriti koji napadi su izvršeni na sustavu. Zastavice nisu u inače prepoznatljivom formatu, već su ujedno i ključni tragovi u forenzičkoj istrazi.

HTB uz to organizira i nekoliko CTF-ova kroz godinu u klasičnom obliku *„Opasnost“*. Jedan od takvih CTF-ova, koji se organizira svake godine, je *HTB Business CTF* koji je namijenjen IT profesionalcima.

Pretplatnicima HTB-a pruža se doživotni pristup svim postojećim slikama strojeva, edukativnim materijalima i sustavu certificiranja. Sljedeća slika prikazuje naslovnu stranicu HTB-a.



Slika 4: *Hack the box* stranica [33]

Slika prikazuje sučelje platforme Hack The Box. Središnji dio prikazuje opcije: VPN Connection za preuzimanje VPN-a, Machines s preko 150 ranjivih strojeva, Challenges za izazove bez VPN-a, te Pro Labs za napredne korisnike.

Na dnu su prikazani podaci o lokaciji VPN-a (engl *Virtual private network*) igrača s najviše veza iz SAD-a (28,37%), te „Attack Map“ s globalnim lokacijama napada na sustav. S desne strane nalazi se top lista timova, s „TheEliteTeam“ na vrhu.

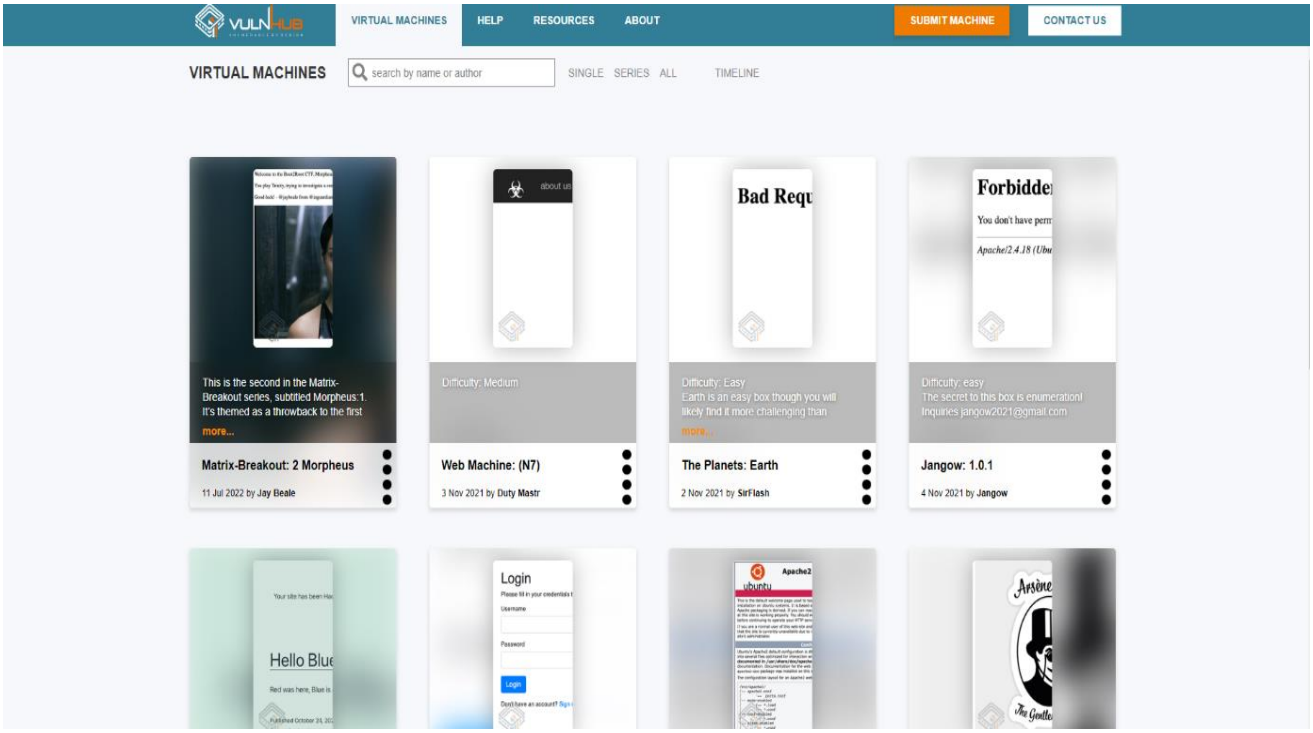
Lijeva strana nudi navigaciju kroz različite opcije, poput edukacije, rangiranja i društvenih mreža.

8.2. VulnHub

VulnHub je projekt i platforma otvorenog tipa gdje svatko može predati svoj *Boot to root CTF* i tako ga učiniti javno dostupnim CTF-ovi se mogu poslati na službeni *X* ili *Facebook* profil. *VulnHub* nema natjecateljsku komponentu, već je njihov cilj pružiti korisnicima što više resursa za učenje etičkog hakiranja kroz praksu [34].

Platforma je posebno korisna za one koji žele poboljšati svoje vještine u području sigurnosti informacijskih sustava jer im ona omogućuje da preuzimaju i rješavaju izazove u vlastitom okruženju. Na taj način se korisnicima pruža prilika za dubinsko istraživanje različitih vrsta ranjivosti i metoda napada, što je ključno za praktično učenje [34].

Kako bi se koristili *VulnHub* izazovi, korisnici trebaju postaviti virtualno okruženje služeći se alatima poput *VirtualBox-a* ili *VMware-a*. Većina CTF-ova na *VulnHubu* dolazi u obliku „*.ova*“ datoteka koje se mogu jednostavno uvesti u ranije spomenute alate za virtualno okruženje. Nakon toga, korisnici su slobodni unaprjeđivati svoje vještine u sigurnom i legalnom okruženju, što je idealno za „etičke hakere“ koji žele izbjeći ilegalne aktivnosti [34]. Sljedeća slika (Slika 5) prikazuje naslovnu stranicu navedene platforme:



Slika 5: VulnHub naslovna stranica [35]

9. Sustav za upravljanje CTF-ovima

U komponentu praktičnog dijela rada razvijen je sustav za upravljanje CTF-ovima *Upravitelj CTF natjecanja* - (engl. *CTF Manager*). Sustav ima dvije uloge. Korisnik koji je igrač CTF-a te administrator koji njime upravlja. Svako CTF natjecanje ima: naslov, opis, datum početka i datum završetka. Igrači koji žele sudjelovati u natjecanju dužni su se na isto prijaviti. Svako natjecanje ima više zadataka. Zadatak se sastoji od naziva, opisa, kategorije i težine koja može biti *lako*, *srednje* i *teško*, zatim broja bodova koje donosi točno rješenje, zastavice koja signalizira točno rješenje i pomoći koja usmjerava igrača pri rješavanju zadatka. Pristup pomoći se bilježi te ju korisnik smije zatražiti svakih 15 minuta unutar jednog natjecanja no svaki zadatak ima samo jednu pomoć. Nakon završetka rješavanja pojedinog zadatka, korisnik može predati PDF dokument koji opisuje način na koji je riješio zadatak. Ti dokumenti su nakon završetka natjecanja dostupni svim korisnicima i služe za učenje.

Aplikacija je pisana u *Laravel* programskom okviru te za razvoj koristi *Laravel-ov* bogat repozitorij biblioteka. Slika 6 prikazuje naslovnu stranicu aplikacije.

Slika 6: Naslovna stranica CTF upravitelja (vlastita izrada)

9.1. Laravel

Laravel je okvir s potpunim tehnološkim stogom (engl. *Technology stack*) pisan u PHP-u. Namijenjen je za izradu aplikacija po MVC paradigmi (engl. *Model-View-Controller*). MVC uzorak je često korišten u razvoju aplikacija. Njegova glavna osobina je podjela aplikacije u tri sloja: model, pogled i upravljač.

Model je sloj odgovoran za interakciju s bazom podataka. *Laravel* za interakciju s modelima koristi *Eloquent*. Za kreiranje migracija koristi se *Iluminate/Migrations* paket.

Pogled je odgovoran za korisničko sučelje koje korisnik vidi. Postoji nekoliko rješenja za prezentacijski sloj. *Blade* je sustav za predloške (engl. *Templating engine*) *Laravela*. On omogućava interpolaciju vrijednosti i uvjeta unutar HTML koda. Sve stranice pripremaju se na poslužitelju i šalju korisniku. Kod promjene stanja stranice, ista će se ponovno učitati. *Livewire* je drugi način prikazivanja prezentacijskog sloja. Omogućava reaktivnost stranice pisanjem programskog koda na strani poslužitelja. Služi kao alternativa za reaktivne vizualne biblioteke za programere fokusirane na stranu poslužitelja. Zadnji način prikazivanja prezentacijskog sloja je korištenjem *Inertiajs-a*. On pruža integraciju *Laravela* s modernim okvirima i bibliotekama za korisničko sučelje kao što su: *React*, *Vue* i *Svelte*.

Za provođenje logike koriste se upravljači. Oni su smisleni skup funkcija koji se brinu za manipulaciju podataka i prikazivanje točnih pogleda [36].

9.2. Modeli i migracije

Modeli u *Laravelu* sastoje se od dvije ključne komponente - migracije i modeli (engl. *Migrations and models*). Migracije su odgovorne za pripremu baze podataka za rad, odnosno kreiranje potrebnih tablica te veza između njih, dok su modeli odgovorni za operacije pretraživanja te čitanja, kreiranja, ažuriranja i brisanja (engl. *Create Read Update Delete - CRUD*) operacija u bazi podataka.

9.2.1. Migracije

Migracije su način za definiranje sheme baze podataka kroz programski kod. Prednost takve definicije je dobra integracija sa sustavima za verzioniranje kao što je *Git*. *Laravel* daje fasadu (jednostavno sučelje za pristup složenom sustavu) zvanu *Shema*. *Fasade u Laravelu* su statičke klase za interakciju s dijelovima sustava. *Shema fasade* služi za interakciju s bazom podataka. Kroz naredbeni redak može se definirati nova migracija. Migracija je klasa koja

nasljeđuje baznu klasu *Migration*. Ima dvije metode: *up* i *down*. *Up* metoda koristi se kod izvršavanja migracije, dok se *down* metoda koristi kod poništavanja migracije [37]. U nastavku se može vidjeti primjer migracije koja kreira tablicu zadataka:

```
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
use App\Models\Natjecanje;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('zadatak', function (Blueprint $table) {
            $table->id();
            $table->string('naslov');
            $table->string('opis');
            $table->string('kategorija');
            $table->string('tezina');
            $table->integer('bodovi');
            $table->string('zastavica');
            $table->foreignIdFor(Natjecanje::class)-
>constrained('natjecanje');
        });
    }
    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('zadatak');
    }
};
```

U programskom kodu iznad može se vidjeti migracija za tablicu „zadatak“. Prikazuju se javne metode „*up*“ i „*down*“ i u njima manipulacije s bazom pomoću *Shema fasade*. Kreira se tablica i zadatak te se pomoću anonimne funkcije dodaju potrebna polja: naslov, opis, kategorija, težina, bodovi i zastavica. Uz to se dodaje i vanjski ključ na tablicu „natjecanje“ kako bi se označila pripadnost zadatka natjecanju. Ta pripadnost bolje je vidljiva na modelima. Javna metoda „*down*“ ima suprotni efekt metodi „*up*“ i briše tablicu „zadatak“.

9.2.2. Modeli

Eloquent je *Laravelov* ORM (engl. *Object-relation mapper*). ORM je biblioteka koja mapira svaku tablicu u bazi podataka na jednu klasu te omogućuje interakciju s tablicom kroz metode na toj klasi bez korištenja SQL-a. Također sadrži i „mehanizam za pristup nižem sloju“ (engl. *Escape hatch*) koji omogućuje korištenje SQL-a za jako kompleksne upite ili pak za upite kod kojih su jako bitne performanse [38].

U nastavku se može vidjeti programski kod modela:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;
use App\Models\Zadatak;

class Natjecanje extends Model
{
    use HasFactory;


    protected $table = 'natjecanje';
    public $timestamps = false;
    protected $fillable = [
        'naslov',
        'opis',
        'pocetak',
        'kraj'
    ];

    protected $hidden = [
```

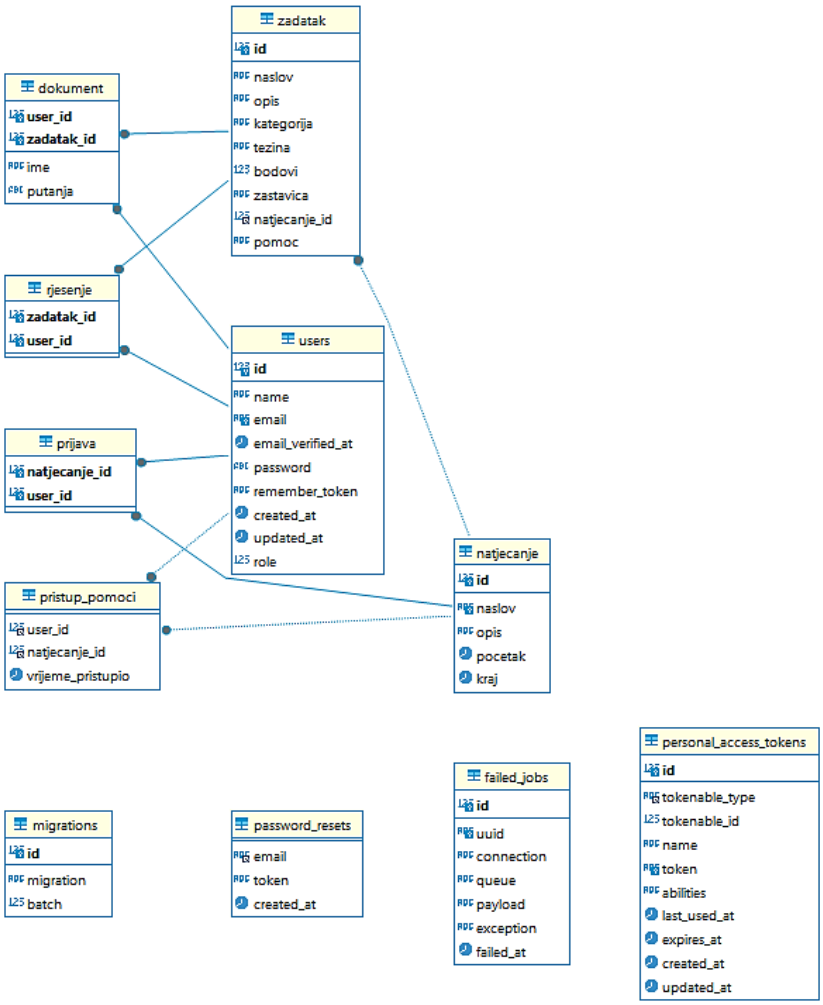
```
    ];

    /**
     * The attributes that should be cast.
     *
     * @var array<string, string>
     */
    protected $casts = [
        'pocetak' => 'datetime',
        'kraj' => 'datetime'

    ];

    protected $guarded = [

    ];
    public function traje()
    {
        $sad = now();
        return $this->pocetak <= $sad && $this->kraj >= $sad;

    }
    public function jeZavrsilo(){
        $sad = now();
        return $sad > $this->kraj;
    }

    public function zadatci(): HasMany
    {

        return $this->HasMany(Zadatak::class);

    }

}
```

Na početku modela definirane su varijable za konfiguraciju: ime tablice na koju se veže model, *bool* vrijednost koja označava želi li se u bazu spremati vrijeme kreiranja i mijenjanja nekog retka, niz vrijednosti koje se smiju popunjavati kako bi se zaštitili od ranjivosti masovnog popunjavanja (engl. *Mass-assignment*). Zatim su definirane vrijednosti koje je potrebno sakriti

kod serijalizacije da ne bi došlo do „curenja“ podataka te vrijednosti koje se trebaju reformatirati tijekom prikaza. Nadalje se može uočiti metoda „traje“, ona je primjer dodavanja poslovne logike unutar modela, tako da vraća *true* ako natjecanje trenutno traje i *false* ako ne traje. Slično je i s metodom „jeZavrsilo“. Metoda „zadaci“ stvara vezu jedan na više sa zadacima kroz metodu *HasMany* bazne klase *Model*. Moguće je izraziti sve vrste veza, čak i one koje koriste među-tablicu za povezivanje.

9.2.3. ERA model aplikacije

Sljedeća slika prikazuje ERA model aplikacije „CTF upravitelj„:



Slika 7: ERA model CTF upravitelja (vlastita izrada)

Tablica „natjecanje“ predstavlja CTF natjecanja. Ona sadrži naslov, opis početak i kraj. Natjecanje u sebi može imati više zadataka, a svaki zadatak mora imati naslov, opis zadatka, kategoriju, težinu koja može biti *lako, srednje i teško,* broj bodova koje donosi, zastavicu koja

signalizira „rješenje“, natjecanje kojem pripada (vanjski ključ na tablicu „natjecanje“) i pomoć. Svaki zadatak može biti rješavan od strane više korisnika, o tome vodi računa slabi entitet „rjesenje“. Da bi mogao rješavati zadatke, korisnik se prvo mora prijaviti na natjecanje. U jednom trenu korisnik može biti prijavljen na više natjecanja. Ako korisnik ne zna nastaviti s rješavanjem zadatka, može pristupiti pomoći. Ta informacija pohranjuje se u tablici „pristup_pomoci“. Svaki zadatak ima svoju pomoć, no pristupi se prate na razini natjecanja kako bi se osigurao period zabrane. Takav period zabrane traje 15 minuta od posljednjeg pristupa pomoći na razini natjecanja. Nakon što uspješno riješi zadatak korisnik može predati rješenje. To prati tablica „dokument“, a svaki korisnik može predati samo jedno rješenje po zadatku.

Tablica *password_reset* služi za resetiranje lozinki. Ona je kreirana od strane biblioteke za autentikaciju *Breeze*. Tablica „*failed_jobs“ je* sustavska tablica u *Laravelu*. Funkcionalnost poslova nije korištena u ovom radu. Tablica „*personal_access_token“* je tablica koja dopušta autorizaciju za pristup API-u aplikacije no ta mogućnost također nije korištena u ovom radu. Tablica „*migration*„ je sustavska tablica metapodataka koja služi za čuvanje migracija koje su se izvršile kako bi se očuvao pravilan redoslijed.

9.3. Putanje i upravljači

Upravljači su poslužiteljska strana *Laravela*. To su klase koje u sebi sadrže logiku za svu obradu podataka, interakciju s bazom podataka i ostalim vanjskim servisima (ako postoje). Putanje su način na koji strana klijenta komunicira s poslužiteljem. Jedna metoda upravljača ili jedan upravljač povezuje se s putanjom koja poziva poslužitelja [39].

9.3.1. Putanje

Kako bi se objasnio koncept putanje dobro je pogledati prvo jednostavni primjer putanje:

```
use Illuminate\Support\Facades\Route;

Route::get('/greeting', function () {
    return 'Pozdrav svijetu';
});
```

Ovo je jednostavni primjer putanje u kojem se koristi statička klasa *Route* na kojoj se poziva statička metoda, u ovom slučaju „*get*“ jer je cilj da to bude metoda zahtjeva. Kao prvi argument daje se putanja na koju se želi da funkcija bude vezana, a drugi argument je

anonimna funkcija koja putanji daje funkcionalnost. U ovom slučaju anonimna funkcija, koja služi kao upravljač, samo vraća „Pozdrav svijetu“ [39].

Sljedeći primjer bit će primjer kompleksnije putanje:

```
Route::post('/natjecanje/{natjecanje}/zadatak/{zadatak}/upload',
[ZadatakController::class, 'upload'])-
>name('natjecanje.zadatak.upload');
```

Ova putanja je tipa „*post*“ što znači da služi za kreiranje. Kroz URL prima parametre „*natjecanje*“ i „*zadatak*“ koji su naznačeni vitičastim zagradama. Drugi argument metode je niz koji sadrži ime metode unutar upravljača koju treba pozvati kako bi izvršili funkcionalnost putanje. Pomoću uzorka dizajna *Builder* dodaje se ime putanje. Ime putanje je način na koji strana klijenta aplikacije može referencirati putanju bez da piše puni put.

9.3.2. Upravljači

Upravljač su načini za grupiranje funkcija u smislene cjeline. Na primjer upravljač „*Korisnik*“ obrađuje sve zahtjeve vezane za korisnika i brine se za prikazivanje „*Pogleda*“ vezanih za korisnika. Svaka metoda unutar upravljača i dalje ima svoju putanju. U nastavku je prikazan jednostavan upravljač i njegovo mapiranje na putanju [40].

```
<?php
namespace App\Http\Controllers;
use App\Models\User;
use Illuminate\View\View;
class UserController extends Controller
{
    /**
     * Show the profile for a given user.
     */
    public function show(string $id): View
    {
        return view('user.profile', [
            'user' => User::findOrFail($id)
        ]);
    }
}
use App\Http\Controllers\UserController;

Route::get('/user/{id}', [UserController::class, 'show']);
```

Ovdje se može vidjeti jednostavan primjer upravljača koji iz putanje prima „*id korisnika*“ te pomoću ORM-a pronalazi tog korisnika i prikazuje ga u pogledu. Prvi parametar je putanja

zajedno s varijabilnim „*id*“ dijelom dok je drugi argument niz koji se sastoji od klase upravljača i metode unutar tog upravljača koju treba pozvati.

Upravljači s jednom akcijom su upravljači koji u sebi imaju samo jednu metodu i rade samo jednu radnju. Tada se ta jedna metoda mora zvati „*__invoke*“ te se upravljači mogu definirati u putanji bez specificiranja metode [40].

9.3.2.1. Resursni upravljači

Laravel svaki model unutar *Eloquenta* smatra „resursom“, a u *Laravelu* nad resursima se ponavljaju četiri CRUD operacije. On zbog toga nudi apstrakciju koja se zove upravljač resursa i u sebi ima sve metode za obavljanje ovih operacija. Dakle, sadrži putanje za poglede kao i za API pozive koje pogledi pozivaju. Resursni upravljač pruža putanju „*index*“ za pregled svih resursa tog tipa (npr. slika). Zatim pruža metode „*create*“ za klijenta i „*store*“ za poslužitelja za operaciju kreiranja. Nakon toga pruža metodu „*show*„ za prikaz pojedinačnog resursa, obično s više detalja. Metode „*edit*“ i „*update*“ služe za poslužiteljske i klijentske akcije uređivanja dok metoda „*destroy*“ služi za brisanje [40].

9.3.2.2. Singleton upravljači

Singleton upravljači su upravljači koji služe za samo jedan resurs i taj resurs je drugačiji za svakog korisnika. Na primjer, „*profile/show*“ i „*profile/edit*“ odnose se na prikazivanje i uređivanje profila trenutno prijavljenog korisnika koji nema pravo uređivati druge profile [40].

9.3.3. Posrednik

Posrednik (engl. *Middleware*) je način za pred procesiranje zahtjeva, npr. provjeru autentifikacije i autorizacije, pisanje u dnevnik i slično. Radi na način da preuzme zahtjev, izvrši svoju funkciju te pošalje zahtjev do upravljača [41]. U nastavku je prikazan jednostavan posrednik koji se koristi za provjeru administratorske uloge kod prijavljenog korisnika. Taj *Posrednik* se koristi za osiguranje putanja kojima samo administratori smiju pristupiti.

```
class JeAdminMiddleware
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure(\Illuminate\Http\Request):
(\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse)  $next
     * @return
\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse
     */
```

```
    public function handle(Request $request, Closure $next)
    {
        if (!Auth::check() || !Auth::user()->jeAdmin()) {
            abort(403, 'Unauthorized action.');
        }
        return $next($request);
    }
}
```

Posrednik provjerava ako korisnik nije prijavljen ili ako mu uloga nije admin. U tom slučaju vraća grešku '403 - Nedozvoljena akcija'. Ako korisnik je administrator, zahtjev dolazi nepromijenjen do kontrolera.

Ispod je prikazan dio programskog koda koji unutar konstruktora upravljača spaja „*auth*“ i „*je_admin*“ posrednike na sve akcije upravljača osim *index* i *show*.

```
class NatjecanjeControler extends Controller
{
    public function __construct(){
        $this->middleware(['auth','je_admin'])->except('index','show');
    }
...
```

9.3.4. Validatori

Svaki zahtjev u sebi sadrži metodu „*validate*“. Ta metoda osigurava da se poslani podaci pridržavaju traženog formata. U nastavku je primjer validatora za CTF zadatke:

```
$validatedData = $request->validate([
      'naslov' => 'required|max:255',
      'opis' => 'required',
      'kategorija' => 'required|max:255',
      'tezina' => 'required|in:lako,srednje,tesko',
      'zastavica' => 'required|max:255',
      'bodovi' => 'required|integer|min:0',
      'pomoc' => 'required|max:255',
      'natjecanje_id' => 'required|exists:natjecanje,id'

  ])
```

U ovom validatoru se može vidjeti obavezno postojanje naslova čija dužina može biti maksimalno 255 znakova. Opis također mora postojati. Mora postojati i težina koja ima jednu

od tri moguće vrijednosti - *lako*, *srednje* ili *teško*. Minimalni iznos bodova mora biti nula te se atribut „*natjecanje*“ mora vezati za postojeće natjecanje budući da je to vanjski ključ [42].

Validacija je vrlo bitan proces u čuvanju integriteta sustava i njegovih podataka. Jako je bitno da su svi podaci sukladni poslovnim pravilima aplikacije [42].

9.4. Pogledi

Pogledi su način za prikazivanje korisničkog sučelja unutar MVC uzorka dizajna. U slučaju CTF upravitelja poglede prikazuju namjenski (engl. *Dedicated*) upravljači. Pogledi su pisani u *Blade* sustavu za predloške, dok su dinamičke komponente pisane u *Livewire* okviru. Sve komponente stilizirane su pomoću *Bootstrap CSS* okvira. Za prikazivanje poruka obavijesti koristi se biblioteka *Toastr* te se na kraju koristi *Vite* kao alat za izgradnju klijenta.

9.4.1. Blade

Blade je sustav za predloške napisan za *Laravel* okvir [43]. U sljedećem dijelu programskog koda može se vidjeti upravljač zadužen za prikazivanje pogleda svih natjecanja.

```
public function index()
{
    $natjecanja = Natjecanje::all();
    return view('natjecanje.index', compact('natjecanja'));
 }
```

Prvo se pomoću *Eloquenta* dohvate sva natjecanja. Zatim se pozove metoda „*view*“ kojoj se kao prvi argument šalje pogled koji se želi prikazati, a kao drugi argument šalje se niz podataka koje se želi proslijediti pogledu. U ovom slučaju taj niz ima samo jedan argument, to je lista svih natjecanja [43].

Blade omogućava da se zajedno piše HTML, CSS, *JavaScript* i PHP programski kod, taj programski kod kompajler kasnije pretvara u PHP.

```
@extends('layouts.app')

@section('content')
<div class="container mt-4">
    @auth
        @if (Auth::user()->jeAdmin())



    <div class="row">
```

```
        <div class="col-12 mb-3">
            <a href="{{ route('natjecanje.create') }}" class="btn btn-
primary"> Kreiraj natjecanje</a>
        </div>
    </div>
    @endif
    @endauth
    <div class="row">

        @foreach ($natjecanja as $natjecanje)
            <div class="col-md-4">
                <div class="card mb-4 shadow-sm">
                <a href="{{ route('natjecanje.show', $natjecanje->id)
}}" class="card-custom-link">
                    <div class="card-body">
                        <h5 class="card-title">{{ $natjecanje->naslov
}}</h5>
                        <p class="card-text">{{
Illuminate\Support\Str::limit($natjecanje->opis, 255) }}</p>
                        <div class="d-flex justify-content-between
align-items-center">
                            <small class="text-muted">Početak: {{
$natjecanje->pocetak->format('d.m.Y H:i') }}</small>
                            <small class="text-muted">Kraj: {{
$natjecanje->kraj->format('d.m.Y H:i') }}</small>
                        </div>
                        </a>
                    </div>
                </div>
            </div>
        @endforeach
    </div>
</div>
@endsection

@section('styles')
<style>
    .card-custom-link {
        color: inherit;
        text-decoration: none;
    }
    .card-custom-link:hover {
        color: inherit;
```

```
            text-decoration: none;
        }
    </style>
    @endsection
```

Blade ima ugrađene posebne *instrukcije*, npr. instrukcija „*extends*“ označava da je trenutni pogled (dijete pogleda navedenog u direktivi) i da ga nadopunjava. To znači da će trenutni pogled tijekom koraka kompilacije biti stavljen unutar roditeljskog pogleda. Pogled je u roditeljskom pogledu pozvan pomoću direktive „@yield“ [43].

```
<main class="py-4">
            @yield('content')
        </main>
```

Također postoji i instrukcija „*auth*“ koja dopušta pristupanje korisničkom modelu za trenutno prijavljenog korisnika. U ovom slučaju provjerava se ima li trenutno prijavljeni korisnik ulogu admina. Samo u slučaju da je uvjet točan prikazuje mu se gumb za kreiranje natjecanja.

Nakon toga prelazi se na glavni dio ovog pogleda. Pomoću „*@foreach*“ *Blade* direktive ispisuju se sva natjecanja. Natjecanja se ispisuju u obliku kartica. Stilove kartica implementira okvir *Bootstrap*. U predlošku se također može vidjeti i nekoliko linija CSS-a koje su lokalizirane samo na ovu datoteku. Ove linije poništavaju predefinirane stilove poveznica u preglednicima [43].

9.4.2. Livewire u Laravelu

Livewire je inovativni programski okvir koji pojednostavljuje izradu dinamičnih i reaktivnih korisničkih sučelja. U *Laravel* aplikacijama kombinira poslužiteljski PHP s klijentskim *JavaScriptom*. Ovaj pristup omogućuje razvoj složenih interaktivnih komponenti bez potrebe za pisanjem opsežnog prilagođenog *JavaScript* koda [44].

9.4.2.1. Princip rada Livewire-a

Prilikom učitavanja, *Livewire* komponenta se renderira kao HTML s ugrađenim JSON metapodacima koji opisuju njeno trenutno stanje (svojstva, metode, itd.). Taj HTML se šalje pregledniku i prikazuje korisniku. *Livewireova JavaScript* biblioteka zatim pretražuje stranicu, inicijalizira komponente i postavlja upravljače događaja (engl. *Event listener*) za korisničke interakcije, kao što su klikovi ili slanje obrazaca [44].

9.4.2.2.Obrada korisničkih interakcija

Kada korisnik stupi u interakciju s komponentom, *Livewire* presreće događaj i šalje AJAX zahtjev *upravljaču događaja* umjesto da ga obrađuje u *JavaScriptu*. Ovaj zahtjev

uključuje trenutno stanje komponente i akciju koju je korisnik pokrenuo. Na strani poslužitelja, *Livewire* ažurira stanje komponente dok PHP programski kod izvršava potrebne radnje, poput ažuriranja baze podataka [44].

9.4.2.3. Ažuriranje komponente

Nakon obrade zahtjeva, poslužitelj ponovo renderira komponentu kako bi odražavao njezino novo stanje i šalje natrag ažurirani HTML zajedno s novim podacima o stanju. *Livewireova JavaScript* biblioteka zatim prima ovaj odgovor i virtualno ažurira DOM, ažurirajući samo dijelove koji su izmijenjeni zbog interakcije korisnika. Ovo osigurava besprijekorno i učinkovito korisničko iskustvo, čuvajući unose u poljima i pozicije trake za pomicanje (engl. *Scroll bar*) bez potrebe za ponovnim učitavanjem cijele stranice [44].

9.4.2.4. Kuke životnog ciklusa i događaji

Osim toga, *Livewire* nudi kuke životnog ciklusa (engl. *Life cycle hooks*) i događaje, koji programerima pružaju detaljnu kontrolu nad različitim fazama životnog ciklusa komponente. Primjer su: učitavanje (engl. *Mount*), ažuriranje i oslobođenje (engl. *Un-mount*). Komponente također mogu emitirati i slušati događaje omogućujući komunikaciju između različitih dijelova aplikacije.

9.4.2.5. Prednosti korištenja Livewire-a

Livewireov pristup smanjuje potrebu za opsežnim prilagođavanjem *JavaScripta* koristeći široke mogućnosti *Laravelovog* poslužiteljskog koda za učinkovito upravljanje dinamičnim i interaktivnim sadržajem. Ova integracija olakšava programerima izgradnju i održavanje složenih korisničkih sučelja bez žrtvovanja performansi ili korisničkog iskustva [44].

Na ovaj način, *Livewire* omogućava jednostavniju izradu modernih web aplikacija kombinirajući prednosti poslužiteljskog PHP-a i klijentskog *JavaScripta*, čineći razvoj bržim i efikasnijim.

Sljedeći programski kod prikazat će reaktivnu komponentu za funkciju pomoći i objasniti njezinu funkcionalnost.

```
<?php

namespace App\Http\Livewire;
use App\Models\PristupPomoci;
use Auth;
use Carbon\Carbon;
use Livewire\Component;
```

```
class Pomoc extends Component
{
    public $zadatak;
    public $pomocOtvorena = false;
    public $odgovor = "";
    public $loading = false;
    public function klikniPomoc()
    {
        $this->loading = true;
        $this->pomocOtvorena = !$this->pomocOtvorena;

        if ($this->pomocOtvorena) {
            $userId = Auth::id();
            $natjecanjeId = $this->zadatak->natjecanje->id;

            $zadnjiPristup = PristupPomoci::where('user_id', $userId)
                                        ->where('natjecanje_id',
$natjecanjeId)
                                        ->latest('vrijeme_pristupio')
                                        ->first();
            if ($zadnjiPristup) {
                $trenutnoVrijeme = Carbon::now("Europe/Zagreb");
                $vrijeme_zadnjeg_pristupa =
Carbon::parse($zadnjiPristup->vrijeme_pristupio,"Europe/Zagreb");

                $razlika = ($trenutnoVrijeme-
>diffInMinutes($vrijeme_zadnjeg_pristupa));



                if ($razlika >= 15) {
                    $this->odgovor = $this->zadatak->pomoc;
                    $pomoc= new PristupPomoci();
                    $pomoc->user_id=  auth()->user()->id;
                    $pomoc->natjecanje_id= $this->zadatak->natjecanje-
>id;
                    $pomoc->save();

                } else {
                    $this->odgovor ="Pomoć će biti dostupna u ".
$vrijeme_zadnjeg_pristupa->addMinutes(15)->format('d.m.Y H:i:s');

                }
```

```
            } else {
                $this->odgovor = $this->zadatak->pomoc;
                $pomoc= new PristupPomoci();
                $pomoc->user_id=  auth()->user()->id;
                $pomoc->natjecanje_id= $this->zadatak->natjecanje->id;
                $pomoc->save();

            }


        }
        $this->loading = false;
    }
    public function mount($zadatak)
    {
        $this->zadatak = $zadatak;
    }
    public function render()
    {
        return view('livewire.pomoc');
    }
}
```

Programski kod otvara „pomoć“ ako je korisniku u tom trenutku omogućeno pravo pristupa i zapisuje taj trenutak pristupa. Ako je korisnik već iskoristio to pravo, ispisuje mu poruku kada će sljedeći put moći pristupiti pomoći. Zadatak dolazi iz argumenata komponente. Pomoć prvotno nije otvorena te je zbog toga odgovor prazan. Stanje „*loading*“ pomaže aplikaciji pratiti je li trenutno u procesu otvaranja ili zatvaranja. Pri otvaranju „pomoći“ utvrdi se posljednji pristup korisnika istoj te ako je to bilo unutar proteklih 15 minuta, ispisuje mu se poruka da treba pričekati. Metoda „*mount*“ prima ulazne argumente dok metoda „*render*“ ispisuje pogled i poziva se kod svake promjene. *Crono* biblioteka koristi se za računanje i formatiranje vremena.

Sljedeća slika prikazuje primjer pomoći:

Slika 8: Primjer pomoći (vlastita izrada)

9.4.3. Bootstrap

Bootstrap je najpopularniji okvir *Kaskadnih stilskih listova (CSS)* i interaktivnih komponenti. Baziran je na HTML-u i CSS-u. *Bootstrap* pruža tipografiju, gotove komponente obrazaca, gumbe, tablice sa straničenjem, sortiranjem i filtriranjem, navigacijske trake i izbornik harmonike. *Bootstrap* također osigurava i responzivan dizajn, prvenstveno fokusiran na mobilne uređaje (engl. *Mobile first*). Ta odluka donesena je zbog veće kompleksnosti responzivnog dizajna na ekranima manjeg formata [45].

```
<div class="accordion" id="accordionPomoc{{$zadatak->id}}">
    <div class="accordion-item">
        <h2 class="accordion-header" id="heading-{{$zadatak->id}}">
            <button class="accordion-button collapsed" type="button"
data-bs-toggle="collapse"
                    data-bs-target="#collapse-{{$zadatak->id}}" aria-
expanded="false"
                    aria-controls="collapse-{{$zadatak->id}}"
wire:click="klikniPomoc" wire:loading.attr="disabled">
                Pomoć
            </button>
        </h2>
        <div id="collapse-{{$zadatak->id}}" class="accordion-collapse
collapse"
             aria-labelledby="heading-{{$zadatak->id}}"
             data-bs-parent="#accordionPomoc{{$zadatak->id}}">
            <div class="accordion-body">
                @if ($pomocOtvorena)
                    {{$odgovor}}
                @endif

            </div>
        </div>
    </div>
</div>
```

Iznad je vidljiv primjer *Bootstrap* izbornika harmonike koji ima mogućnost prikazivanja i sakrivanja, a gumb dodaje i miče atribut „*accordion-colapse*“ i tako kontrolira stanje izbornika harmonike.

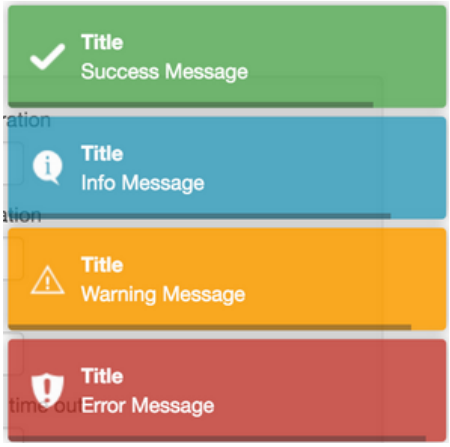
Bootstrap je najpopularniji okvir za korisnička sučelja. Trenutno su aktualne „komponente bez zaglavlja" (engl. *Headless*) koje pružaju interaktivnost bez stilova. Također, neke biblioteke komponenti poput *RadixUI-a* imaju praksu uključivanja samo onih komponenti koje su potrebne preko naredbenog retka. Činjenica da je svaka komponenta samostalna i samoodrživa potiče programere da preuzmu vlasništvo nad komponentama i mijenjaju ih po potrebi.

9.4.4. Toastr

Toastr je implementacija *PHPFlasher* biblioteke za *Laravel*. *PHPFlasher* je biblioteka koja ima direktnu integraciju s dva najveća PHP okvira - *Laravel* i *Symfony*. Instalira se pomoću sljedećih naredbi:

```
composer require php-flasher/flasher-toastr-laravel
php artisan flasher:install
```

Prva naredba instalira PHP biblioteku, dok druga naredba dodaje *Flasher u Vite* konfiguraciju. Moguće je prikazati obavijest uspjeha, informacije, upozorenja i greške [46] [47]. Sljedeća slika prikazuje sve tipove poruka.



Slika 9: Poruke s obavijestima [48]

Sljedeći programski kod primjer je poziva skočne poruke nakon ispravnog kreiranja natjecanja. Skočni prozor poziva se odmah prije preusmjeravanja i pokazuje se na sljedećoj stranici:

```
toastr()->success("Natjecanje kreirano");
    return redirect()->route('natjecanje.index');
```

9.4.5. Vite

Vite je moderni alat za izradu *JavaScript* projekata koji nudi brzu kompilaciju, izuzetno brzu zamjenu komponenata (engl. *Hot Module Replacement - HMR*) uz minimalnu konfiguraciju. Razvijen od strane Evana Youa, kreatora *Vue.js*. *Vite* koristi ES module (ESM) za razvoj, što omogućuje preglednicima da izravno učitavaju module iz datotečnog sustava. Ova metoda eliminira potrebu za korakom pred izgradnje (engl. *Pre-built*) rezultirajući znatno bržim pokretanjem poslužitelja u usporedbi s tradicionalnim alatima poput *Webpacka* [49] [50].

Jedna od ključnih prednosti *Vite-a* je njegova sposobnost brze obrade modula. Ta pogodnost je omogućena zahvaljujući korištenju „*esbuilda*“ koji je pisan u *Go* jeziku i omogućava pred izgradnju ovisnosti (engl. *Pre-bundling*) do 100 puta brže nego alati bazirani na *JavaScriptu*. *Vite* također koristi alat *Rollup* za produkcijska kompajliranja osiguravajući optimizirane datoteke spremne za distribuciju [49] [50].

Vite podržava širok spektar *JavaScript* okvira kao što su *Vue, React, Preact, Lit, Svelte* i mnogi drugi. Ima ugrađenu podršku za *TypeScript*, *CSS* i *JSON uvoženja (*engl*. Import)*, što dodatno pojednostavljuje razvojne procese. *Vite* također nudi bogat skup proširenja, temeljenih na *Rollupovoj plugin API* infrastrukturi, omogućavajući korisnicima lako proširivanje funkcionalnosti [51] [52]. Osim toga, *Vite* se odlikuje izuzetno brzim HMR-om, koji omogućava gotovo trenutačne promjene u kodu bez potrebe za ponovnim učitavanjem cijele stranice. Ovo značajno poboljšava produktivnost programera jer smanjuje vrijeme čekanja tijekom razvoja aplikacija. *Vite* također automatski upravlja dijeljenjem CSS koda i optimizacijom učitavanja asinkronih dijelova, čime se dodatno povećava učinkovitost [51] [52]. Uz ove karakteristike, *Vite* nudi jednostavnu konfiguraciju i fleksibilnost, što ga čini idealnim alatom za moderne web projekte, uključujući i one koji koriste napredne razvojne paradigme kao što su SPA i PWA (engl. *Progressive Web Apps*) koje mogu funkcionirati bez pristupa internetu [50] [53].

9.5. Breeze autentifikacija

Laravel Breeze je jednostavna biblioteka koja pruža kostur za autentifikaciju u *Laravelu*. Biblioteka kreira migracije, modele i poglede kako bi podržala autentifikaciju. Omogućuje prijavu, registraciju, promjenu lozinke i potvrdu e-maila. *Breeze* se može koristiti sa sva tri moguća načina kreiranja pogleda s *Blade-om, Livewire-om i Inertiajs-om*. Biblioteka se instalira kroz naredbeni redak pomoću sljedeće naredbe [54]:

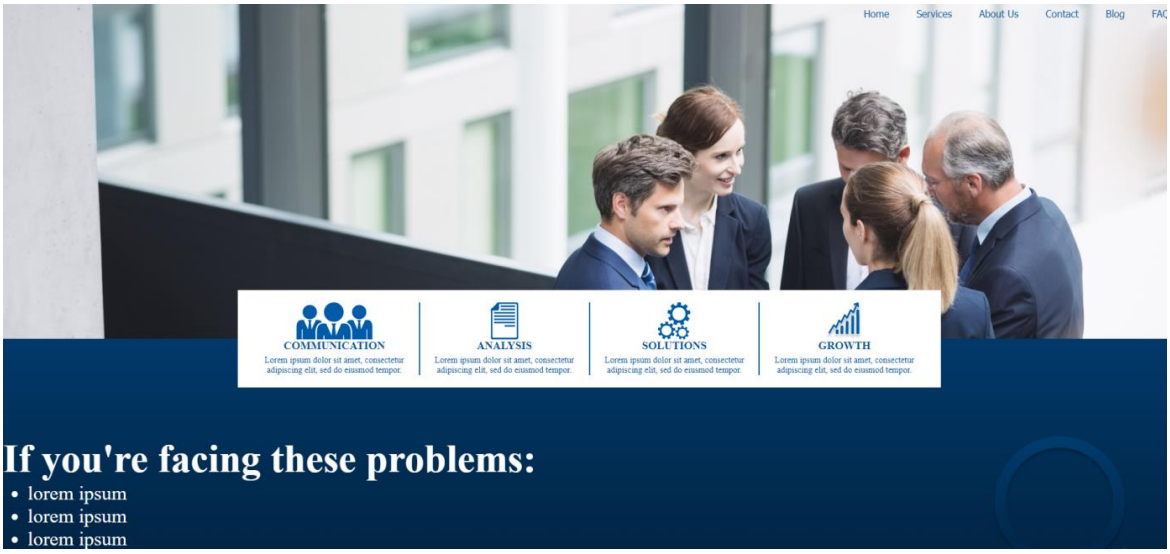
```
php artisan breeze:install
```

Ta naredba kreira sve potrebne kontrolere, modele i migracije. Potrebno je samo izvršiti migracije pomoću naredbe: `php artisan migrate`.

10. CTF aplikacija

CTF aplikacija je slična Web mjestu neke kompanije. Njezina uloga je da uključuje izazove koje natjecatelji moraju riješiti. Sadrži ukupno 5 izazova: izazov pronalaženja zastavice u slici, odnosno steganografiju, krađa sesije pomoću XSS napada. Nakon ulaska u administratorsku ploču grubom silom sljedeći izazov je SQL napad za eskalaciju privilegija, zatim slijedi izazov ubacivanja naredbi (engl. *Command injection*), taj napad daje pristup operacijskom sustavu kao nisko-privilegiranom korisniku korištenjem ljuske. Na kraju korisnik dolazi do posljednjeg izazova u kojem je potrebno enumerirati sustav te pronaći i iskoristiti grešku u konfiguraciji za dobivanje „*root*“ ljuske.

Sljedeća slika prikazuje početnu stranicu CTF aplikacije:



Slika 10: Početna stranica CTF aplikacije (vlastita izrada)

10.1. Metode i tehnike izrade

Natjecanje je izrađeno korištenjem HTML-a, CSS-a i PHP-a. Većina stranica su statičke HTML i CSS stranice, a dinamičke komponente su izrađene pomoću PHP-a. Za upravljanje sesijom koristi se „Klasa za upravljanje sesijama“ preuzeta s predmeta „Web dizajn i programiranje“ [55]. Klasa pruža jednostavniju apstrakciju sučelja korištenog u PHP-u te je to razlog njenog korištenja.

Za bazu podataka koristi se baza podataka *SQLite*. Izabrana je zbog svoje jednostavnosti za postavljanje i korištenje.

Aplikacija se nalazi unutar *VirtualBox* virtualnog stroja. Konfiguracija virtualnog stroja napravljena je pomoću alata *Vagrant*. *Vagrant* je alat za konfiguriranje virtualnih strojeva kroz *Ruby* programski jezik.

Također, kako bi krađa sesije bila moguća, korištena je „*Pyppeteer*“ skripta. Skripta simulira korisnika s moderatorskim pravima koji posjećuje stranicu za čitanje korisničkih poruka te tako postaje žrtva XSS napada koji mu krade sesiju.

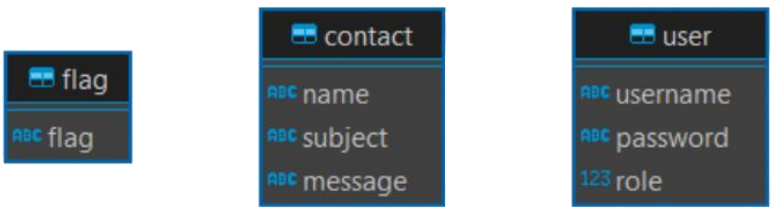
10.2. HTML i CSS

Klijentska strana cijele web aplikacije napisana je korištenjem HTML-a i CSS stilova. Koriste se vrlo bazični elementi kao što su zaglavlja, odjeljci i blokovi. Pozicioniranje stranice napravljeno je ili korištenjem osobine tijeka elemenata ili korištenjem modela kutije.

U stilovima se upotrebljavaju bijela boja i razne nijanse plave boje zajedno s gradacijom.

10.3. Baza podataka - SQLite

Baza podataka sadrži tablicu poruka poslanih administratorskom timu, tablicu koja sadrži korisnike u administratorskom timu (administratora i moderatora) i tablicu koja sadrži zastavu.



Slika 11: ERA model baze CTF-a (vlastita izrada)

SQLite je mala, brza, pouzdana i jednostavna relacijska baza podataka koja implementira većinu standardnog SQL jezika. Razvijena je kao samoodrživi sustav, što znači da sve svoje funkcionalnosti uključuje unutar jedne biblioteke bez potrebe za dodatnim komponentama. *SQLite* je "*serverless*“, što znači da ne zahtijeva pokretanje poslužitelja za

rad, već pohranjuje sve podatke u jednoj datoteci na disku. Ova baza podataka je također "*zero-configuration*", što znači da ne zahtijeva nikakvu konfiguraciju [56].

Zbog svoje jednostavnosti i učinkovitosti, *SQLite* se primjenjuje u širokom rasponu aplikacija, od malih ugrađenih sustava do velikih web usluga. Često se koristi kao ugrađena baza podataka u mobilnim uređajima, web preglednicima i raznim drugim aplikacijama. *SQLite* je sustav otvorenog koda (engl. *Open source*) koji je dostupan besplatno za komercijalnu i nekomercijalnu upotrebu, bez potrebe za licencom. Njegov programski kod je raspoloživ za preuzimanje i prilagodbu, što omogućuje korisnicima da ga modificiraju prema svojim specifičnim potrebama [56].

Jedna od ključnih prednosti *SQLite-a* je njegova visoka pouzdanost (engl. *Reliability*), koja je rezultat opsežnog testiranja i rigorozne kontrole kvalitete. *SQLite* je također poznat po svojoj maloj veličini i visokoj efikasnosti, što ga čini idealnim za ugrađene sustave i aplikacije koje zahtijevaju učinkovito upravljanje podacima [56].

10.4. PHP

PHP je skriptni jezik za programiranje na strani poslužitelja. U CTF aplikaciji zbog male veličine i kompleksnosti aplikacije koristi se u kombinaciji s HTML-om i CSS-om za interakciju s bazom. Također se koristi za autentifikaciju i autorizaciju. Za te događaje uvijek se koristi programski jezik na strani poslužitelja zbog smanjene mogućnosti manipulacije [57].

Sljedeća skripta unosi poruke korisnika u bazu podataka.

```
<?php

$dbPath = 'sqlite:./ctf.db';

try {

    $db = new PDO($dbPath);
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);



    $name = $_POST['name'] ?? '';
    $subject = $_POST['subject'] ?? '';
    $message = $_POST['message'] ?? '';

    $sql = "INSERT INTO contact (name, subject, message) VALUES (:name,
:subject, :message)";
```

```
    $stmt = $db->prepare($sql);

    $stmt->bindParam(':name', $name);
    $stmt->bindParam(':subject', $subject);
    $stmt->bindParam(':message', $message);

    $stmt->execute();

    echo "Red uspiješno unesen.";
} catch (PDOException $e) {
    echo "Error: " . $e->getMessage();
}


$db = null;

header("Location: ./index.html");
exit;
?>
```

Još jedna skripta korištena u CTF aplikaciji je skripta administratorske ploče i u nastavku će biti prikazani samo njezini bitni dijelovi. Na početku skripte provjerava se je li korisnik prijavljen, odnosno ima li valjanu sesiju. Ako ima, dohvaća kontakte. Kontakti se mogu filtrirati po imenu osobe koja ga kontaktira. Taj prvi dio skripte služi za autentifikaciju administratora i moderatora te je vidljiv u nastavku:

```
<?php

require_once ("Sesija.class.php");

$username = $_POST['username'] ?? '';
$password = $_POST['password'] ?? '';

$dbPath = 'sqlite:./ctf.db';


try {
    $pdo = new PDO($dbPath);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

```
    $stmt = $pdo->prepare('SELECT * FROM user WHERE username =
:username');
    $stmt->execute([':username' => $username]);
    $user = $stmt->fetch(PDO::FETCH_ASSOC);

    if ($user && password_verify($password, $user['password'])) {
        $sesija = new Sesija();
        $sesija->kreirajKorisnika($user["username"],$user["role"]);
        header("Location: ./admin-panel.php");
    } else {
        echo "login fail";
        header("Location: ./index.html");
    }
} catch (PDOException $e) {
    die("Could not connect to the database: " . $e->getMessage());
}
?>
```

Ova skripta dohvaća potencijalnog člana administratorskog tima s imenom poslanim u POST zahtjevu. Pomoću metode „*password_verfy*“ provjerava se je li njegova lozinka jednaka onoj spremljenoj u bazi. Ako jest, za administratora ili moderatora kreira sesiju i time mu daje potrebna prava. U slučaju pogrešne lozinke, korisnika se šalje na naslovnu stranicu.

Sljedeći dio skripte administratorske ploče, u slučaju dostatnih prava, prikazuje dohvaćanje i filtriranje kontakata:

```
<?php
require_once ("Sesija.class.php");

if (Sesija::dajKorisnika() == null) {
    echo "Session create fail";
    header("Location: ./admin-login.html");
}

$dbPath = 'sqlite:./ctf.db';

try {
    $db = new PDO($dbPath);
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    if (isset($_GET["filter"])) {
        $filter=$_GET["filter"];
```

```
        $sql="SELECT * FROM contact WHERE name = '" . $filter . "'";
        $result = $db->query($sql);

    } else {
        $sql = "SELECT * FROM contact";
        $result = $db->query($sql);
    }
```

Može se primijetiti da upit nije napisan na siguran način jer se ne koriste pripremljeni izrazi (engl. *Prepared statements*). To omogućava napad *SQL* ubacivanja.

Niže navedeni dio skripte administratorske ploče u tabličnom obliku prikazuje sve ranije dohvaćene zapise. Sadržaj zapisa dolazi od korisnika, što znači da se može pretpostaviti da dolazi iz nesigurnog izvora, no na njemu nije provedena sigurnosna provjera.

```
    <div class="contact-table">
                <table>
                    <tr>
                        <th>Name</th>
                        <th>Tiitle</th>
                        <th>Message</th>
                    </tr>
                    <?php
                    foreach ($result as $row) {
                        echo "<tr>";
                        echo "<td>" . $row['name'] . "</td>";
                        echo "<td>" . $row['subject'] . "</td>";
                        echo "<td>" . $row['message'] . "</td>";
                        echo "</tr>";
                    }
                    ?>
                </table>
                <?php
} catch (PDOException $e) {
    echo "Error: " . $e->getMessage();
}
$db = null;
?>
```

Sljedeći dio programskog koda također pripada skripti administratorske ploče, no dostupan je samo administratorima. Ovaj programski kod služi za provjeru statusa dostupnosti poslužitelja. Programski kod poziva komandu „*system*“ koja izvršava naredbe naredbenog retka. Poziva se naredba *ping* koja šalje (engl. *Internet Control Message Protocol- ICMP*)

pakete udaljenom poslužitelju. Ispis naredbe se zanemaruje, bitan je samo izlazni kod. Ako je izlazni kod 0, naredba je uspješna. Ako izlazni kod ima bilo koju drugu vrijednost, naredba nije uspjela.

```
<?php
  if (Sesija::dajKorisnika()["uloga"] == 2) {
      echo '<div class="forma">
      <form action="admin-panel.php" , method="post">
          <label for="ip">IP:</label>
          <input type="text" name="ip" value="127.0.0.1">
          <button type="submit">Send</button>
      </form>

  </div>';
  }
  ?>

  <div class="output">
      <h1>
          <?php
          if (isset($_POST["ip"]) and Sesija::dajKorisnika()["uloga"]
== 2) {
              $command = 'ping -c 1 ' . $_POST["ip"] . ' > /dev/null
2>&1';

              echo "<pre>";
              system($command, $return_var);
              echo "</pre>";

              if ($return_var === 0) {
                  echo "Health Check Successful: System is online and
can connect to the internet.";
              } else {
                  echo "Health Check Failed: System might be offline
or unable to connect to the internet.";
              }
          }
          ?>
```

Korisnik u polje komande može unijeti unos po želji koji nije filtriran i validiran ni na koji način.

10.5. Automatizirani korisnik

Automatizirani korisnik napisan je pomoću biblioteke *Pyppeteer*. Skripta se pokreće pomoću *cron* zadatka svakih 5 minuta.

10.5.1. Pyppeteer

Pyppeteer je biblioteka napisana u svrhu prevođenja postojeće biblioteka *Puppeteer* iz programskog jezika *JavaScript* u programski jezik *Python*. Biblioteka pruža sučelje za programabilnu interakciju s *Chrome-om* (*Chromium-om*) i drugim preglednicima [58]. Sljedeća skripta koristi biblioteku *Pyppeteer* kako bi simulirala korisnika s privilegijama moderatora.

```
import asyncio
from pyppeteer import launch


async def main():
    browser = await launch(
        headless=True,
        executablePath="/snap/bin/chromium",
        slowMo=5,
        args=['--no-sandbox', '--headless']
    )
    page = await browser.newPage()

    await page.goto('http://localhost/admin-login.html')

    await page.type('input[name="username"]', 'moderator')
    await page.type('input[name="password"]', 'Y0uShallN0tP455!')

    login_button_selector = 'button[type="submit"]'
    await page.click(login_button_selector)
    content = await page.content()
    print(content)
    await browser.close()

asyncio.get_event_loop().run_until_complete(main())
```

Na početku je vidljivo da se uvoze biblioteke *Asyncio* i *Pyppeteer*. *Asyncio* je biblioteka koja pruža asinkrono izvršno okruženje u *Pythonu* pomoću sintakse *async* i *await*. Takva sintaksa se također koristi u programskim jezicima poput *JavaScripta* i *Rusta*.

Najprije se definira asinkrona „*main*“ funkcija, zatim se dohvati objekt preglednika koji ne prikazuje korisničko sučelje (engl. *Headless*) i pokreće se *Chromium* instaliran na virtualnom stroju. Završetak svake akcije čeka se maksimalno 5 sekundi. Preglednik se pokreće i uz argument „*no-sandbox*“ što isključuje njegov 'pješčanik' - jedan od sigurnosnih mehanizama preglednika. Iz objekta preglednika u varijabli „*browser*“ dobiva se objekt stranice. Pomoću metoda na objektu stranice (u kodu vidljivo kao objekt „*page*“) može se interaktivno koristiti web stranicom. Posjećuje se stranica za prijavu. U HTML polja upisuju se korisničko ime i lozinka te se isti šalju na gumb tipa *submit*. Nakon tih akcija simulator se preusmjerava na stranicu s korisničkim porukama i izvršava eventualni *JavaScript* kod koji se tamo nalazi. Zadnja linija gore navedenog programskog koda pokreće „*main*“ funkciju na asinkroni način.

10.5.2. Cron zadaci

Cron zadaci su mehanizam za izvršavanje skriptnih ili kompajliranih programa u određeno vrijeme uz mogućnost ponavljanja. *Cron* zadatke izvršava *Cronov* program u pozadini (engl. *Daemon)* po definiranom rasporedu. Sljedeća komanda u ime „*root“* korisnika postavlja *cron* [59].

```
sudo sh -c '(crontab -u root -l; echo "*/5 * * * * /usr/bin/python3
/home/automated-user/main.py") | crontab -u root -'
```

Pomoću naredbe „*sudo*“ pokreće se naredba u ime korisnika *root. U crontab* se zapisuje da se svakih pet minuta pomoću *Pythona* pokreće skripta automatiziranog korisnika. Prva zvjezdica označava minute što znači bilo koja minuta, a „/5 „označava vremenski interval od 5 minuta, odnosno program se pokreće svakih 5 minuta. Daljnja mjesta nisu bitna za ovaj raspored. Ona redom predstavljaju: sat, dan u mjesecu, mjesec i dan u tjednu.

10.6. Virtualizacija

CTF radi unutar sustava za virtualizaciju *VirtualBox-a*, a izgrađen je pomoću *Vagranta* sustava za izgradnju virtualnih strojeva kroz *Ruby* kod.

10.6.1. VirtualBox

VirtualBox je program otvorenog programskog koda. On nudi pokretanje virtualnih operacijskih sustava koji imaju svoje virtualne komponente unutar korisnikovog računala. *VirtualBox* je najpoznatiji alat za virtualizaciju. Uz njega postoje još i alati poput *VMWare-a i Qemu-a* [60].

10.6.2. Vagrant

Vagrant je alat za konfiguraciju virtualnih strojeva kroz programski kod podržanih tipova virtualnih strojeva, pa tako i *VirtualBox-a*. Služi kao deterministično konfiguriranje virtualnih strojeva. Dijeli mnogo prednosti s ostalim alatima za podizanje infrastrukture kroz programski kod. Zbog toga što je konfiguracija napisana u programskom kodu i automatizirana, omogućuje višestruko ubrzanje konfiguracije te otklanja mogućnost ljudske greške, što znači da će se iz iste *Vagrant* konfiguracijske datoteke uvijek podići isti virtualni stroj [61].

U nastavku je prikazan programski kod za pokretanje virtualnog stroja za CTF:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :


Vagrant.configure("2") do |config|

  config.vm.box = "ubuntu/focal64"
  config.vm.network "public_network"


  config.vm.synced_folder ".", "/vagrant", disabled: true


     config.vm.provider "virtualbox" do |vb|

      vb.gui = true


       vb.memory = "4096"
       vb.cpus="4"
       vb.name="CTF"
     end
     config.vm.provision "file", source: "./www/", destination: "/tmp/"
```

```
     config.vm.provision "file", source: "./automated-user/",
destination: "/tmp/automated-user"


  config.vm.provision "shell", inline: <<-SHELL
    sudo loadkeys hr
    apt-get update
    apt-get install -y apache2
    sudo mv  /tmp/www/* /var/www/html/
    sudo touch /var/www/html/ctf.db
    sudo chown -R www-data:www-data /var/www/html
    sudo chmod -R 755 /var/www/html
    sudo apt-get install software-properties-common
    sudo add-apt-repository ppa:ondrej/php
    sudo apt-get update
    sudo apt-get install php8.2  libapache2-mod-php8.2 -y
    sudo apt install sqlite3 -y
    sudo apt install php8.2-sqlite3 -y
    sudo sqlite3 /var/www/html/ctf.db "CREATE TABLE contact(name text,
subject text, message text);"
    sudo sqlite3 /var/www/html/ctf.db "CREATE TABLE user(username text,
password text, role integer);"
    PLAINTEXT_PASSWORD="mrroboto"
    HASHED_PASSWORD=$(php -r "echo password_hash('$PLAINTEXT_PASSWORD',
PASSWORD_BCRYPT);")
    sudo sqlite3 /var/www/html/ctf.db "INSERT INTO user VALUES
('admin','$HASHED_PASSWORD',2);"
    PLAINTEXT_PASSWORD="Y0uShallN0tP455!"
    HASHED_PASSWORD=$(php -r "echo password_hash('$PLAINTEXT_PASSWORD',
PASSWORD_BCRYPT);")
    sudo sqlite3 /var/www/html/ctf.db "INSERT INTO user VALUES
('moderator','$HASHED_PASSWORD',1);"
    sudo sqlite3 /var/www/html/ctf.db "CREATE TABLE flag(flag text);"
    FLAG="CTF{223797ae419b4189128f2895c44252c8}"
    sudo sqlite3 /var/www/html/ctf.db "INSERT INTO flag VALUES
('$FLAG');"
    sudo service apache2 start
    sudo ufw allow 80
    yes | sudo ufw enable
    sudo service apache2 restart
    sudo update-rc.d apache2 defaults
```

```
    sudo mv /tmp/automated-user /home/
    sudo chown -R root:root /home/automated-user
    sudo apt-get update
    sudo apt-get install -y python3-pip
    snap install chromium
    sudo -H pip3 install -r /home/automated-user/requirements.txt
    sudo sh -c '(crontab -u root -l; echo "*/5 * * * * /usr/bin/python3
/home/automated-user/main.py") | crontab -u root -'
    echo 'www-data ALL=(ALL) NOPASSWD: /usr/bin/awk' > /tmp/www-data-awk
    visudo -c -f /tmp/www-data-awk && mv /tmp/www-data-awk
/etc/sudoers.d/
    sudo mkdir -p /home/www-data/
    sudo echo "CTF{d60b3309622970d9151d521e75f1d6a5}" >/home/www-
data/flag4.txt
    sudo chown -R www-data:www-data /home/www-data/flag4.txt
    sudo echo "CTF{35a5b0fb193f83e480ab8153f0471cdd}" >/home/flag5.txt
    sudo chown -R root:root /home/flag5.txt
    echo 'vagrant:Dadasada123' | sudo chpasswd
  SHELL
End
```

Prethodni kod konfigurira virtualni stroj na bazi *Ubuntu Focal* distribucije. Stroj dobiva 4 gigabajta RAM-a, 4 jezgre procesora i pristup lokalnoj mreži. Pomoću upravljačkog programa na virtualni stroj su prebačene datoteke za web aplikaciju i automatiziranog korisnika. Jezik tipkovnice prebačen je na hrvatski. Sustav je ažuriran te su instalirani programi *SQLite*, PHP, *Python* i *Apache web poslužitelj*. U bazi su postavljeni korisnici te je dodana jedna zastavica. Kroz zaštitnu stijenu dopušten je samo port 80. Dozvoljeno je da korisnik *www-data* pokrene program *awk* kao *root korisnik* kako bi se dopustila eskalacija privilegija.

11. Zaključak

CTF (Engl. *Capture the flag*) je natjecanje u rješavanju zadataka iz raznih područja informacijske sigurnosti kao što su *Web*, *kriptografija*, *steganografija*, *reverzno inženjerstvo, Pwn* i *Boot-to-root*. Završni rad u teorijskom dijelu opisu razvoj alata za učenje informacijske sigurnosti u obliku CTF natjecanja te teorijske osnove informacijske sigurnosti. Ne opisuju se osnove programiranja te osnove mreža i mrežnih protokola koji su nužni za razumijevanje sadržaja. Rad opisuje tipove ranjivosti putem OWASP top 10 liste, kao i metode za njihovu prijavu. Objašnjavaju se najčešće kategorije zadataka na CTF natjecanjima. U sklopu rada razvijena je platforma za upravljanje CTF-ovima kao i CTF natjecanje.

U praktičnom djelu rada CTF upravitelj daje mogućnost organiziranja CTF-a. On kroz svoje mogućnosti dopušta administraciju natjecateljske komponente CTF-a te je napisan pomoću *Laravela*, jednog od najpoznatijih PHP okvira. CTF upravitelj dozvoljava kreiranja CTF natjecanja i zadataka s težinama: lako, srednje i teško unutar njih. Igračima pruža mogućnost prijave na natjecanja i rješavanja zadataka unutar vremena održavanja. Igrači nakon prijave imaju pravo zaigrati CTF u vremenu održavanja. CTF upravitelj predviđa samostalnu igru, dok platforme poput *Hack the box-a* ponekad dopuštaju timsku Unutar cjelokupnog natjecanja postoji prilika za korištenjem pomoći svakih 15 minuta, no to korištenje je ograničeno na jednu pomoć unutar pojedinog zadatka. Igrači kao dokaz rješenja zadatka predaju zastavicu te za to dobiju bodove. Bodovi su vidljivi na top listi. Igrači također mogu predati dokument koji pokazuje rješenje zadatka. Ostali igrači rješenjima mogu pristupiti po završetku natjecanja. Izrađeno CTF natjecanje u svrhu rada je pisano u PHP programskom jeziku i pokreće se unutar *VirtualBox-a* čija je konfiguracija pisana u *Vagrantu*. CTF sadrži 5 zadataka: otkrivanje stenografske poruke unutar slike, krađu sesije putem XSS-a, podizanje prava putem SQL ubacivanja, izvršavanje naredbi nad poslužiteljskim računalom putem ubacivanja naredbi i podizanje privilegija zbog nesigurne konfiguracije. Ovaj skup ranjivosti omogućava korištenje raznih tehnika snimanja infrastrukture te jedan mogući put za iskorištavanje ranjivosti. Težina iskorištavanja ranjivosti unutar CTF natjecanja prilagođena je početnicima.Rad objašnjava *ExploitDB* i OWASP top 10 listu ranjivosti kao metode za pronalazak ranjivosti, odnosno kao izvore za pronalazak ranjivosti.

Predstavljene su mogućnosti za nastavak učenja informacijske sigurnosti kroz platforme *VulnHub* i *Hack the box*.

CTF pokazuje jedan od načina za učenje informatičke sigurnosti koja je važna za sve informatičare, a ne samo za one koji se specijaliziraju ili su specijalizirani za sigurnost informacijskih sustava.

Popis literature

[1] ctftime.org, „What is Capture The Flag?“ Pristupljeno: 13. svibanj 2024. [Na internetu]. Dostupno na: https://ctftime.org/ctf-wtf/

[2] ibm.com, „What is a Cyberattack? | IBM“. Pristupljeno: 06. rujan 2024. [Na internetu]. Dostupno na: https://www.ibm.com/topics/cyber-attack

[3] N. Dissanayake i K. Dias, „Web-based Applications: Extending the General Perspective of the Service of Web“, svi. 2017.

[4] D. V Kornienko, S. V Mishina, i M. O. Melnikov, „The Single Page Application architecture when developing secure Web services“, *J Phys Conf Ser*, sv. 2091, izd. 1, str. 12065, stu. 2021, doi: 10.1088/1742-6596/2091/1/012065.

[5] Kenneth Lange, „Don’t Limit Your REST API to CRUD Operations“. Pristupljeno: 13. svibanj 2024. [Na internetu]. Dostupno na: https://kennethlange.com/dont-limit-your-rest-api-to-crud-operations/

[6] R. D. Hartley i D. Medlin, „Ethical Hacking : Educating Future Cybersecurity Professionals“, 2017. [Na internetu]. Dostupno na: https://api.semanticscholar.org/CorpusID:198181906

[7] owasp.org, „Vulnerability Disclosure Cheat Sheet“. Pristupljeno: 14. svibanj 2024. [Na internetu]. Dostupno na: https://cheatsheetseries.owasp.org/cheatsheets/Vulnerability_Disclosure_Cheat_Sheet.html#vulnerability-disclosure-cheat-sheet

[8] kali.org, „Kali Linux“. Pristupljeno: 16. svibanj 2024. [Na internetu]. Dostupno na: https://www.kali.org/docs/nethunter/

[9] parrotsec.org, „What is ParrotOS? | ParrotOS Documentation“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://parrotsec.org/docs/introduction/what-is-parrot/

[10] owasp.org, „OWASP Top 10:2021“. Pristupljeno: 27. srpanj 2024. [Na internetu]. Dostupno na: https://owasp.org/Top10/

[11] owasp.org, „A01:2021 – Broken Access Control“. Pristupljeno: 16. svibanj 2024. [Na internetu]. Dostupno na: https://owasp.org/Top10/A01_2021-Broken_Access_Control/

[12] owasp.org, „A02:2021 – Cryptographic Failures“, Pristupljeno: 16. svibanj 2024. [Na internetu]. Dostupno na: https://owasp.org/Top10/A02_2021-Cryptographic_Failures/

[13] owasp.org, „A03 Injection - OWASP Top 10:2021“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://owasp.org/Top10/A03_2021-Injection/

[14] owasp.org, „A04 Insecure Design - OWASP Top 10:2021“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://owasp.org/Top10/A04_2021-Insecure_Design/

[15] owasp.org, „A05 Security Misconfiguration - OWASP Top 10:2021“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/

[16] owasp.org, „A06 Vulnerable and Outdated Components - OWASP Top 10:2021“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/

[17] owasp.org, „A07 Identification and Authentication Failures - OWASP Top 10:2021“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/

[18] owasp.org, „A08 Software and Data Integrity Failures - OWASP Top 10:2021“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/

[19] owasp.org, „A09 Security Logging and Monitoring Failures - OWASP Top 10:2021“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/
[20] owasp.org, „A10 Server Side Request Forgery (SSRF) - OWASP Top 10:2021“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/
[21] exploit-db.com, „ Exploit Database Submissions Guidelines “. Pristupljeno: 17. svibanj 2024. [Na internetu]. Dostupno na: https://www.exploit-db.com/submit
[22] atan, „What is CTF and how to get started!“, dev.to. Pristupljeno: 18. svibanj 2024. [Na internetu]. Dostupno na: What is CTF and how to get started!
[23] OSIRIS Lab i CTFd, „Web Exploitation“, ctf101.org. Pristupljeno: 18. svibanj 2024. [Na internetu]. Dostupno na: https://ctf101.org/web-exploitation/overview/
[24] charcharbinks.com, „CTF Cryptography for Beginners“. Pristupljeno: 18. svibanj 2024. [Na internetu]. Dostupno na: https://charcharbinks.com/post/ctf_crypto_for_beginners/
[25] packetlabs.net, „What is a Cryptographic Attack? Your Comprehensive Guide“. Pristupljeno: 18. svibanj 2024. [Na internetu]. Dostupno na: https://www.packetlabs.net/posts/what-is-a-cryptographic-attack/
[26] ctf101.org, „Steganography“. Pristupljeno: 24. lipanj 2024. [Na internetu]. Dostupno na: https://ctf101.org/forensics/what-is-stegonagraphy/
[27] fareedfauzi.gitbook.io, „Steganography“. Pristupljeno: 18. svibanj 2024. [Na internetu]. Dostupno na: https://web.archive.org/web/20240605003157/https://fareedfauzi.gitbook.io/ctf-playbook/steganography
[28] ctf101.org, „Reverse Engineering“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://ctf101.org/reverse-engineering/overview/
[29] ctf101.org/, „Forensics“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://ctf101.org/forensics/overview/
[30] ctf101.org, „Binary Exploitation“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://ctf101.org/binary-exploitation/overview/
[31] BioFrosted, „boot2root vs ctf? : r/HowToHack“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://www.reddit.com/r/HowToHack/comments/whksbt/boot2root_vs_ctf/
[32] Ryan Gordon, „Introduction to Hack The Box “. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://help.hackthebox.com/en/articles/5185158-introduction-to-hack-the-box
[33] Rhett Greenhagen, „Best websites to help you build your hacking skills. | by Rhett Greenhagen | Medium“. Pristupljeno: 28. srpanj 2024. [Na internetu]. Dostupno na: https://rgreenhagen.medium.com/best-websites-to-help-you-build-your-hacking-skills-75f4246a9ea1
[34] vulnhub.com, „About VulnHub“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://www.vulnhub.com/about/
[35] vulnhub.com, „Vulnerable By Design ~ VulnHub“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://www.vulnhub.com/
[36] laravel.com, „Installation - Laravel 10.x - The PHP Framework For Web Artisans“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://laravel.com/docs/10.x/installation
[37] laravel.com, „Database: Migrations“. Pristupljeno: 20. svibanj 2024. [Na internetu]. Dostupno na: https://laravel.com/docs/10.x/migrations

[38] laravel.com, „Eloquent: Getting Started - Laravel 10.x - The PHP Framework For Web Artisans“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://laravel.com/docs/10.x/eloquent
[39] laravel.com, „Routing - Laravel 10.x - The PHP Framework For Web Artisans“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://laravel.com/docs/10.x/routing
[40] laravel.com, „Controllers - Laravel 10.x - The PHP Framework For Web Artisans“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://laravel.com/docs/10.x/controllers
[41] laravel.com, „Middleware - Laravel 10.x - The PHP Framework For Web Artisans“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://laravel.com/docs/10.x/middleware
[42] laravel.com, „Validation - Laravel 10.x - The PHP Framework For Web Artisans“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://laravel.com/docs/10.x/validation
[43] laravel.com, „Blade Templates - Laravel 10.x - The PHP Framework For Web Artisans“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://laravel.com/docs/10.x/blade
[44] Caleb Porzio, „How Livewire works (a deep dive)“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://calebporzio.com/how-livewire-works-a-deep-dive
[45] w3schools.com, „Bootstrap CDN“. Pristupljeno: 30. svibanj 2024. [Na internetu]. Dostupno na: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp
[46] Younes ENNAJI, „Toastr | PHPFlasher“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://php-flasher.io/library/toastr/
[47] Younes ENNAJI, „Laravel | PHPFlasher“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://php-flasher.io/laravel/
[48] Younes ENNAJI, „GitHub - yoeunes/toastr toastr.js notifications for Laravel“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://github.com/yoeunes/toastr
[49] Tim Severien, „What is Vite? An Overview of the New Front-end Build Tool“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://www.sitepoint.com/vitejs-front-end-build-tool-introduction/
[50] vitejs.dev, „Why Vite | Vite“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://vitejs.dev/guide/why
[51] Eric Simons, „What is Vite (and why is it so popular)?“ Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://blog.stackblitz.com/posts/what-is-vite-introduction/
[52] Codemotion, „ViteJS: A Modern and Powerful Frontend Optimization Tool“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://www.codemotion.com/magazine/frontend/web-developer/vitejs-guide/
[53] Amelie Lamb, „Vite.js - Lightning Fast Builds Powered by ES Modules - DEV Community“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://dev.to/amelie-lamb/vitejs-lightning-fast-builds-powered-by-es-modules-i4n
[54] laravel.com, „Starter Kits - Laravel 10.x - The PHP Framework For Web Artisans“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://laravel.com/docs/10.x/starter-kits
[55] drsc Dragutin Kermek, „Web dizajn i programiranje“, str. 15–20, Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://elfarchive2223.foi.hr/pluginfile.php/11061/mod_resource/content/13/predavanja/Kermek_WebDiziProg_17.pdf

[56] sqlite.org, „About SQLite“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://www.sqlite.org/about.html
[57] php.net, „PHP: What is PHP? - Manual“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://www.php.net/manual/en/intro-whatis.php
[58] checklyhq.com, „What is Puppeteer? | Checkly“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://www.checklyhq.com/learn/headless/basics-puppeteer-intro/
[59] Thinus Swart, „Cron Jobs: The Complete Guide for 2024“, sij. 2024, Pristupljeno: 03. lipanj 2024. [Na internetu]. Dostupno na: https://cronitor.io/guides/cron-jobs
[60] virtualbox.org, „Oracle VM VirtualBox“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://www.virtualbox.org/
[61] hashicorp.com, „Introduction | Vagrant | HashiCorp Developer“. Pristupljeno: 26. lipanj 2024. [Na internetu]. Dostupno na: https://developer.hashicorp.com/vagrant/intro

Popis slika