Projektni zadatak iz predmeta

Informaciona Bezbednost

Školska 2022/2023. Godina

Projekat predstavlja veb aplikaciju za rad sa sertifikatima. Aplikacija se razvija kao monolit, u proizvoljnom steku tehnologija. Celokupno rešenje mora biti dostupno na nekoj platformi za kontrolu verzija. Sam projekat se radi u timovima od 3 studenta. Studenti ne moraju da budu iz iste grupe, samo će za odbranu kontrolnih tačaka ceo tim dolaziti u jednu grupu.

Tipovi korisnika

- **Neautentifikovani korisnik** - Može da se registruje na sistem čime dobija običan nalog. Ako poseduje nalog, može da se prijavi na sistem.
- **Autentifikovani korisnik** - Korisnik koji se prijavio na običan profil. Ima mogućnost da pregleda i preuzme sve sertifikate. Takođe, može da zatraži da mu se izda sertifikat koji će biti potpisan od strane nekog drugog (*root/intermediate*) sertifikata. Može da zatraži da se određeni sertifikat povuče i može da izvrši validaciju bilo kog sertifikata preko ove aplikacije. Korisnik može da bude vlasnik samo *intermediate* ili *end* sertifikata. Može da odobri ili odbije zahteve za izdavanje sertifikata koji se odnose na izdavanje novih sertifikata potpisanih od strane onih koje posmatrani korisnik ima.
- **Admin** - Specijalni korisnik koji ne može da se registruje. Zadužen za upravljanje *root* sertifikatima koje koristi sama aplikacija kao i za odobravanje zahteva za izdavanje sertifikata koji su potpisane od strane *root* sertifikata. Može da uradi sve što i autentifikovani korisnik.

Delovi sistema

- **Local file storage** - Mesto gde će se sami sertifikati čuvati
- **Baza podataka -** SQL/NoSQL baza podataka gde će se čuvati ostali neophodni podaci vezani za korisnike, vlasništvo nad sertifikatima…
- **Backend -** obavezn deo sistema gde će se implementirati logika, dozvoljeno je raditi u bilo kom jeziku i bilo kojoj tehnologiji.
- **Frontend -** nije obavezan i neće se ocenjivati. Pod ovim se smatra da mora postojati neki frontend za prikaz podataka ali se neće ocenjivati izgled, raspored komponenti i tome slično. U koliko tim želi, može koristiti neki *templating engine* za generisanje frontend-a na backendu poput Jinja2 za python, ASP.NET Core MVC za .NET ili Thymeleaf za javu.

Funkcionalni zahtevi

1. **Registracija profila**
    a. Neregistrovani korisnik može da se registruje na sistem tako što će uneti validan email, ime, prezime, broj telefona, lozinku i dodatne podake, u koliko za njima ima potrebe. (2 boda)
    b. Nakon registracije korisnik mora da potvrdi vlasništvo nad makar jednim od unetih resursa (email ili broj telefona) time što će se poslati zahtev za potvrdu identiteta na taj resurs. (2 bod)
2. **Prijava na sistem**
    a. Neregistrovani korisnik može da se, unosom email-a i lozinke prijavi na sistem i tako pristupi funkcionalnostima autentifikovanog korisnika (2 boda)
    b. Neregistrovani korisnik tek nakon potvrde identiteta (email-a ili broja telefona) može da se prijavi na sistem i pristupi funkcionalnostima autentifikovanog korisnika (2 bod)
3. **Oporavak lozinke**
    a. Korisnik koji je zaboravio lozinku može da započne proces oporavka lozinke. Na resurs koji korisnik poseduje (email ili broj telefona) šalje se verifikacioni kod koji će korisnik uneti uz dva puta ponovljenu novu lozinku kako bi izvršio oporavak lozinke. (2 boda)
4. **Rotacija lozinki**
    a. Neophodno je implementirati mehnizam koji tera korisnike da nakon određenog vremena vrše obnavljanje lozinke. Za demonstraciju treba postaviti da lozinke mogu da važe proizvoljan period ali treba razmotriti i *best practice* za ovakav zahtev. Radi jednostavnosti dovoljno je da se pri svakoj prijavi proveri da li je od poslednjeg zanavljanja lozinke prošlo vreme koje je ta lozinka važila, i ako jeste naterati korisnika da lozinku zanovi kako bi mogao da nastavi interakciju sa sistemom. (1 bod)
    b. Treba omogućiti da korisnici ne mogu da zanove lozinku sa nekom od *n* prethodnih lozinki. (1 bod)
5. **Pregled svih sertifikata**
    a. Svim prijavljenim korisnicima treba omogućiti pregled svih sertifikata. U sam pregled treba uključiti podatke o tome kad je sertifikat izdat, kome (kom korisniku) je izdat, tip sertifikata (*root, intermediate, end*). (1 bod)
6. **Preuzimanje sertifikata**
    a. Treba omogućiti korisniku da preuzme sertifikat. Ovo važi i za povučene i istekle sertifikate. (2 boda)
7. **Provera validnosti sertifikata**
    a. Omogućiti korisniku da proveri validnost nekog sertifikata na osnovu identifikatora. (2 boda)

b. Omogućiti korisniku da proveri validnost nekog sertifikata na osnovu upload-ovane kopije sertifikata. (2 boda)

8. **Pravljenje zahteva za novi sertifikat**
    a. Svaki autentifikovani korisnik ima mogućnost da napravi zahteva za izdavanje sertifikata. Korisnik može da zahteva samo *intermediate* ili *end* sertifikate, dok admin može da zahteva i *root* sertifikat. Zahtev se prosleđuje onom korisniku koji je vlasnik sertifikata koji treba da potpiše sertifikat za koji je podnet zahtev. Ako se sertifikat izdaje na osnovu sertifikata čiji je isti korisnik i vlasnik, zahtev se automatski odobrava. Ako se sertifikat izdaje na osnovu *root* sertifikata same aplikacije, admin mora da odobri sertifikat. Ako admin zahteva da se napravi neki sertifikat na osnovu nekog drugog sertifikata ili ako je to novi *root* sertifikat, zahtev se automatski odobrava. (3 boda)
9. **Pregled zahteva**
    a. Autentifikovani korisnik ima uvid u stanje aktivnih i prošlih zahteva za izdavanje sertifikata koje je podneo. (2 boda)
    b. Admin ima uvid u sve aktivne i prošle zahteve za izdavanje sertifikata. (1 bod)
10. **Odobravanje/odbijanje zahteva za sertifikat**
    a. Autentifikovani korisnik, koji je vlasnik sertifikata na osnovu kojeg postoji zahtev za izradu novog sertifikata, ima opciju da određeni zahtev prihvati, ili odbije. Ako prihvati pravi se novi sertifikat i potpisuje se izabranim sertifikatom. Ako odbije, iznosi razlog zbog kojeg se zahtev odbija i proces se završava. (3 boda)
11. **Povlačenje sertifikata**
    a. Autentifikovani korisnik koji je vlasnik nekog sertifikata (izuzev *root* sertifikata) može u bilo kom trenutku da povuče sertifikat uz obrazloženje povlačenja istog. Obratiti pažnju na to da ako je sertifikat povučen, svi u lancu ispod njega se automatski povlače. To znači da svi sertifikati koje je on potpisao kao i sertifikati koje su ti sertifikati potpisali se automatski povlače. (4 boda)
12. **HTTPS komunikacija**
    a. Sama aplikacija treba da koristi *root* sertifikat o kojem će admin voditi računa. Ako se frontend implementira kao zasebna aplikacija (popularni frontend radni okviri poput angular, vue.js ili react koji koriste odvojeni server) neophodno je da i oni koriste isti sertifikat. (1 bod)
    b. Baza podataka treba da koristi odvojeni sertifikat i treba omogućiti da baza i backend mogu da komuniciraju preko enkriptovanih poruka (u zavisnosti od tehnologije moguće je da protokol komunikacije između baze i servera

nije isključivo HTTP(s) tako da se mora omogućiti enkripcija poruka u protokolu koji tehnologija zahteva). (1 bod)

13. **Dvofaktorska autentifikacija**
    a. Kada se korisnik prijavljuje na sistem upotrebom lozinke, treba zatražiti da se potvrdi da je to zaista on tako što će se on dodatno autentifikovati preko verifikacionog koda koji će se poslati na jedan uređaj koji je korisnik izabrao (email ili telefon) (2 boda)
    b. Treba voditi računa o dužini trajanja sesije. Nakon isteka određenog vremenskog perioda treba naterati autentifikovane korisnike da se ponovo autentifikuju. Za demonstraciju je neophodno da ovaj period bude kratak, ali treba istražiti *best practice* metode. (1 bod)
14. **ReCAPTCHA za forme**
    a. Kako bismo zaštitili forme od “spamovanja” treba implementirati mehanizam ReCAPTHCA-e. Mogu se koristiti bilo koja rešenja za koja se tim dogovori (Google-ove ReCAPTCHAv2, ReCAPTCHAv3 ili Cloudflare Turnstile). Naglašava se da je validaciju tokena neophodno uraditi i na klijentskoj i serverskoj strani. (1 bod)
15. **OAuth protokol**
    a. Implementirati prijavu uz pomoć nekog delegiranog načina pristupa. Voditi računa da korisnik, ako ima registrovan nalog (email) i prijavi se uz pomoć OAuth protokola on ne treba da napravi novi nalog već da ga poveže sa postojećim. (2 boda)

Nefunkcionalni zahtevi

1. **Validacija podataka (2 boda)**
    a. Sprečiti relevantne *Injection* napade
    b. Sprečiti *XSS* napade
    c. Sprečiti *Path Traversal* napade
    d. Izvršiti validaciju podataka, koristeći kriterijume validacije definisane po najboljim praksama za pisanje bezbednog koda
    e. Proveriti tipove datoteka koji se upload-uju (u slučaju validacije sertifikata putem uploadovanja), kao i ograničiti veličinu fajlova koji se mogu poslati.
2. **Zaštita podataka (2 boda)**
    a. Osetljivi podaci sa kojima aplikacija radi treba da budu obezbeđeni u skladištu, u transportu i tokom upotrebe. Identifikovati osetljive podatke i implementirati prikladne bezbednosne kontrole. Osetljivi podaci čije se skladištenje ne može izbeći treba da budu šifrovani ili heširani u zavisnosti od prirode podataka.
3. **Monitoring uz logove (2 boda)**

a. Kako log zapisi predstavljaju osnovni mehanizam za postizanje neporecivosti, neophodno je omogućiti njihovo prikupljanje. Od infrastrukture (operativni sistem, baza podataka, …) do alata (radni okvir, biblioteke), značajan deo koda nije pod našom kontrolom, ali to ne smanjuje našu odgovornost ako nam softver bude eksploatisan zbog ranjivosti u nekoj *third-party* komponenti te je neophodno voditi log zapise o tome ko pristupa kojim resursima na našem sistemu. Logove treba implementirati prateći *best practise* standarde. (Za projekat je dovoljno implementirati prikupljanje logova na nivou aplikacije, no studenti se podstiču da istraže kako bi mogli to da urade na nekom višem nivou, na primer na nivou sistema ili više distribuiranih sistema)

4. **Mailserver (2 boda)**
    a. Kako bismo izbegli probleme koje SMTP protokol može da donese, neophodno je implementirati integraciju sa nekim mail serverom koji će za nas da šalje poruke. Primer jednog takvog je Sendgrid, no danas ih ima dosta više.
5. **Ranjivosti (model pretnji) (2 boda)**
    a. Definisati alate koji će se koristiti za proveru različitih skupova komponenti (sqlmap, dependency-check…)
    b. Neophodno je izvršiti provere i napraviti neki vid izveštaja o tome šta je pronađeno i dokumentovati korake koji slede
    c. Analizirati ozbiljnosti ranjivosti i mogućnost eksploatacije
    d. Definisati i izvršiti strategiju za razrešavanje mogućih rizika

Kontrolne tačke

U toku semestra će se održati dve kontrolne tačke nakon kojih sledi finalna odbrana. Same kontrolne tačke će se održati u terminu vežbi po rasporedu koji će naknadno biti objavljen i to:

1. U nedelji od 3.4.2023. do 5.4.2023.
2. U nedelji od 8.5.2023. do 10.5.2023.

Finalna odbrana će biti organizovana u junskom roku i biće subota prepodne (sadašnji termin koji je predložen je 17.6.2023. ali je moguće da će se ovaj datum malo pomeriti). Ako dođe do pomeranja termina, termin će biti pomeren za neki dalji datum u junu. Za dalje rokove kada će se projekat moći braniti bićete blagovremeno obavešteni i u tim rokovima će se bodovi skalirati.

Raspodela posla po kontrolnim tačkama:

- Prva kontrolna tačka (15 bodova):
    - 1a, 2a, 5a, 7a, 8a, 9a, 10a

    - Za ovu kontrolnu tačku ne morate imati nikakav frontend i dovoljno je da demonstrirate funkcionalnost kroz Postman.
    - Takođe, za ovu kontrolnu tačku nije neophodno voditi računa o *best practise* predlozima za kontrole.
- Druga kontrolna tačka (15 bodova):
    - 1b, 2b, 3a, 6a, 7b, 9b, 11a
    - Za ovu kontrolnu tačku je neophodno započeti frontend, i očekuje se da će se minimalno (ili nikako) koristiti Postman.
    - Ovde će se pregledati i *best practise* predlozi za stvari koje su do sad implementirane
- Finalna odbrana (20 bodova):
    - 4a, 4b, 12a, 12b, 13a, 13b, 14a, 15a
    - Očekuje se da na ovoj proveri ne koristite Postman za demonstraciju, već da se sve izvršava kroz grafički interfejs
    - Na ovoj proveri će se pregledati poštovanje *best practise* predloga kao i nefunkcionalni zahtevi
        - Napomena za nefunkcionalne zahteve: Iako se sastoje iz tačaka, bodovanje će biti rađeno po principu “ili sve ili ništa”. Tako na primer da biste dobili sve bodove vezane za tačku *Validacija podataka* morate ispoštovati sve tačke od *a* do *e*.

Bodovi

Projekat nosi 50 poena, i ostalih 50 poena nosi usmeni ispit. Za polaganje predmeta neophodno je da imate 50% bod na oba dela i da je ukupan broj poena >= 51.

Važi pravilo da na jednoj odbrani možete vratiti bodove samo sa prethodne odbrane što znači da ako ste izgubili neke bodove na jednoj kontrolnoj tački možete ih vratiti samo na proveri koja sledi (bilo da je kontrolna tačka ili finalna odbrana). Izuzetak su odbrane projekta u kasnijim rokovima gde nema povraćaja izgubljenih bodova sa kontrolnih tačaka.

Napomena

Iako se broj bodova na odbrani u daljim rokovima skalira i dalje morate ostvariti minimalan broj bodova iz obe oblasti kako biste mogli da položite predmet što znači da će vam biti neophodan veći broj implementiranih stvari za isti broj bodova.

FAQ