SIGURNOSNI ASPEKTI SERVISNO ORIJENTISANIH ARHITEKTURA
SERVICE-ORIENTED ARCHITECTURE SECURITY VULNERABILITIES

Saša Marić, Institut za zaštitu zdravlja Republike Srpske
Zoran Đurić, Univerzitet u Banjoj Luci Elektrotehnički fakultet

Sadržaj – *U ovom radu opisani su najvažniji sigurnosni aspekti servisno orijentisanih arhitektura, te najvažnije tehnike čija primjena omogućava kreiranje sigurnih servisno orijentisanih aplikacija. Pored skupa prednosti koje servisno orijentisana arhitektura (SOA) donosi, nova paradigma sa sobom donosi i niz novih potencijalnih sigurnosnih ranjivosti. Iz tog razloga neophodno je unaprijediti tehnike za obezbjeđivanje sigurnosti aplikacija. U ovom radu dat je kratak osvrt na SOA koncepte, nakon čega su analizirane najvažnije sigurnosne ranjivosti SOA. Posebno su analizirane ranjivosti koje su jedinstvene za SOA, odnosno koje se ne pojavljuju kod drugih arhitektura. Isto tako, u radu su analizirane i tehnike kojima se sigurnosne ranjivosti SOA mogu eliminisati, čime se omogućava kreiranje sigurnih aplikacija.*

Abstract - *This paper describes the most important service-oriented architecture security vulnerabilities and the most important techniques whose usage makes secure service oriented applications creation possible. In addition to a set of benefits that service-oriented architecture (SOA) brings, a new paradigm also brings a number of new potential security vulnerabilities. For this reason, it is essential to upgrade techniques for applications securing. This paper gives a brief overview of SOA concepts, after which the most important SOA security vulnerabilities are analyzed. Specific vulnerabilities that are unique for SOA and do not appear in other architectures are separately analyzed. Furthermore, the paper analyzes techniques that can eliminate SOA security vulnerabilities thus enabling development of secure applications.*

1. UVOD

Postoji nekoliko različitih definicija servisno orijentisane arhitekture (*eng. Service Oriented Architecture,* skraćeno SOA). Ipak, ono što je zajedničko za sve definicije jeste činjenica da servisno orijentisana arhitektura predstavlja posebnu paradigmu za realizaciju i održavanje poslovnih procesa u distribuiranim sistemima [1].

Istorijski gledano, SOA je logičan nastavak tehnološkog razvoja u oblasti informacionih tehnologija kojim se na nov i jednostavniji način pokušava riješiti problem povezivanja heterogenih i distribuiranih sistema. Naime, i do sada su postojale tehnologije kojima je bilo moguće povezati heterogene i distribuirane sisteme, kao što su CORBA (*eng. Common Object Request Broker Architecture*), RMI (*eng. Remote Method Invocation*) i RPC (*eng. Remote Procedure Call*), ali sve su zahtijevale tzv. jaku spregu.

SOA se bazira na tri osnovna koncepta: servisima, interoperabilnosti i slaboj sprezi. U SOA se kao servisi koriste web servisi. Interoperabilnost se postiže korištenjem otvorenog standarda za razmjenu poruka. U ovu svrhu koristi se SOAP (*eng. Simple Object Access Protocol*) protokol. SOAP poruke su specifični XML dokumenti (*eng. eXtensible Markup Language*) koji se razmjenjuju putem platformski nezavisnog SOAP protokola. Konačno, slaba sprega se postiže korištenjem standardizovanog WSDL (*eng. Web Services Description Language*) opisa servisa i UDDI (*eng. Universal Description, Discovery and Integration*) registra, te javnim objavljivanjem servisa na mreži [2].

2. SIGURNOSNI ASPEKTI APLIKACIJA I POJAM SIGURNOSTI

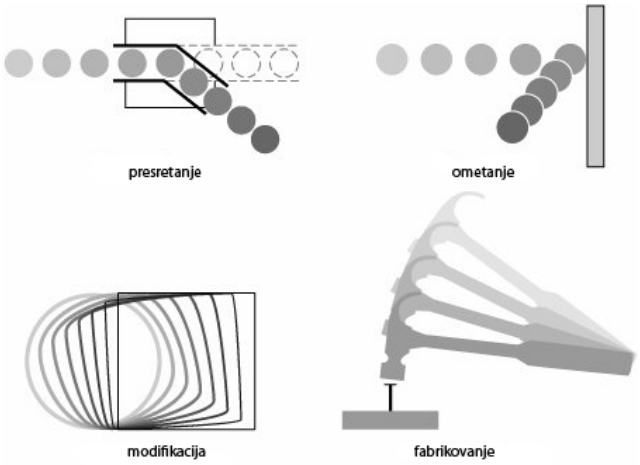
Sigurnost aplikacija jedan je od njihovih najvažnijih aspekata, kako pri njihovom razvoju, tako i u toku njihove eksploatacije. Kada se govori o sigurnosti u oblasti informacionih tehnologija, tada se mogu posmatrati tri razdvojene, ali gotovo podjednako važne komponente: hardver, softver i podaci. Svaka komponenta ima svoju vrijednost u okviru sistema i zahtjeva posebne tehnike zaštite. Osjetljivosti sistema, prijetnje, napadi i kontrola mogu se jednim terminom nazvati sigurnosnim aspektima [3].

Osjetljivost predstavlja sigurnosnu slabost sistema koja postoji u proceduri, dizajnu ili samoj implementaciji, a koja može biti iskorištena za neovlaštenu zloupotrebu sistema. Primjera radi, neki sistem može da bude osjetljiv po pitanju neovlaštene manipulacije podacima zato što nema implementiran mehanizam utvrđivanja identiteta korisnika prije omogućavanja pristupa podacima.

Prijetnja predstavlja skup okolnosti koje potencijalno mogu da naprave štetu sistemu.

Za lice ili sistem koji pokušava da iskoristi ili iskorištava osjetljivost sistema kaže se da vrši *napad na sistem*.

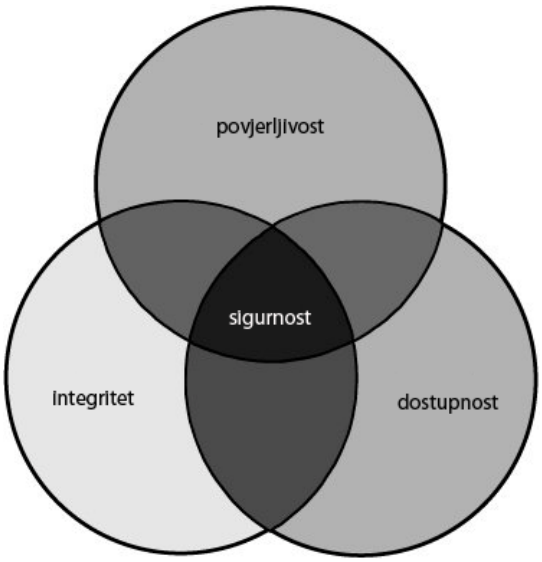
Konačno, *kontrola* je zaštitna mjera odnosno akcija, procedura, uređaj ili tehnika koja uklanja ili u značajnoj mjeri umanjuje osjetljivost sistema, čime onemogućava ili u značajnoj mjeri otežava vršenje napada na sistem, odnosno sistem štiti od prijetnji.



Slika 2.1 – Sigurnosne prijetnje [3]

Generalno, sigurnosne prijetnje mogu da se podijele u četiri grupe (slika 2.1) [3]:

- presretanje – situacija u kojoj neovlaštena strana dobija pristup nekom resursu sistema ili podacima koji se prenose,
- prekid (ometanje) - situacija u kojoj sistem, zbog aktivnosti napadača, postaje nedostupan ili je njegovo korištenje jako otežano za legitimne korisnike sistema,
- modifikacija - situacija u kojoj neovlaštena strana ne samo da ima pristup nekom resursu sistema, već nad tim resursom vrši i promjene, i
- fabrikovanje – situacija u kojoj neovlaštena strana izvrši fabrikaciju, odnosno krivotvorenje nekog resursa sistema koji dalje legitimnim korisnicima predstavlja kao legitiman resurs.



Slika 2.2 – Pojam računarske sigurnosti [3]

Termin sigurnost veoma se često koristi u svakodnevnom životu. Kada se govori o računarskoj ili softverskoj sigurnosti, postoje različite podjele koje identifikuju najvažnije zahtjeve koje jedan sistem mora ispuniti da bi se mogao smatrati sigurnim. Prema jednoj od najčešće primjenjivanih podjela, tri najvažnija sigurnosna aspekta bilo kog informacionog sistema su: povjerljivost, integritet i dostupnost (slika 2.2) [3].

Povjerljivost podrazumijeva garantovanje da resursima mogu pristupiti samo i isključivo samo autorizovani korisnici. To znači da samo oni koji trebaju da imaju pristup nekom resursu zapravo i mogu da mu pristupe, s tim da se pristup ne ograničava na pregledanje, čitanje, mijenjanje i slično, već i otkrivanje postojanja resursa.

Integritet podrazumijeva da resursi mogu da budu promijenjeni samo od strane autorizovanih korisnika ili samo na autorizovan način. U ovom kontekstu, promjena uključuje kreiranje, mijenjanje, promjenu statusa i brisanje resursa.

Dostupnost podrazumijeva da se resursima može pristupiti od strane autorizovanih korisnika u određeno vrijeme. Drugim riječima, ne smije se zabraniti legitiman pristup resursima za koje je korisnik autorizovan.

3. NAJVAŽNIJI SIGURNOSNI ASPEKTI SOA

SOA implementacije danas postoje u gotovo svim granama industrije. Međutim, novom paradigmom i novim pristupom projektovanju i realizaciji povezivanja distribuiranih sistema SOA je donijela novine i u pogledu sigurnosti. Ne samo da se javila potreba za razvojem novih sigurnosnih koncepata i tehnika, već se ispostavilo da su mnoge do sada često korištene sigurnosne prakse u slučaju SOA nedovoljne, pa čak i kontraproduktivne [4].

Za razliku od ranijih distribuiranih sistema, SOA podrazumijeva slabu spregu. Isto tako, klijenti web servisa i web servisi mogu da budu implementirani u različitim jezicima, te distribuirani na različitim platformama. Zbog toga, prethodno veoma popularni centralizovani pristup sigurnosti više nije moguć. Nadalje, SOA implementacija u vidu web servisa, obično podrazumijeva da se komunikacija između klijenata i servisa vrši preko HTTP/HTTPS (*eng. Hypertext Transfer Protocol* i *eng. Hypertext Transfer Protocol Secure*) transportnog protokola, što isto tako može predstavljati dodatni sigurnosni problem [5].

U nastavku su u kratkim crtama opisane najznačajnije osjetljivosti SOA. Neke od opisanih osjetljivosti nisu jedinstvene samo za SOA, već se kao takve javljaju i u drugim arhitekturama, s tim da u SOA imaju dodatne specifičnosti.

XML DoS (*eng. XML Denial of Service,* skraćeno XDoS) – DoS napadi su kategorija napada koju je moguće vršiti nad gotovo svim vrstama aplikacija. Novitet koji je došao uvođenjem XML dokumenta kao osnovnog „elementa“ koji se razmjenjuje u SOA komunikaciji, donio je i novu specifičnu klasu DoS napada poznatu kao XDoS. Naime, procesiranje XML dokumenata može biti kompleksno i procesorski zahtjevno što može dovesti do određenih sigurnosnih problema. Zbog toga, DoS napadi se uglavnom baziraju na loše formiranim i/ili prevelikim porukama čija obrada zahtijeva konzumaciju značajnog dijela resursa servisa [6]. Ako napadač formira XML poruku koja je obimna, sadrži rekurzivne sadržaje, ima previše sadržaja koji su ugniježdeni ili maliciozne DTD-ove (*eng. Data Type Documents*), obrada ovakvih XML poruka može dovesti do preopterećenja procesora, memorije ili komunikacionog kanala, te samim tim i zagušenja web servisa dovodeći do

DoS-a, odnosno situacije u kojoj web servis neće moći prihvatiti niti obraditi zahtjev legitimnog klijenta [7].

Jedna specifična klasa takvih napada je napad takozvanim XML bombama. Naime, jedna od osobina DTD-ova je i ta da DTD ima mogućnost da automatski preuzme i učita entitete definisane unutar DTD-a. Ukoliko napadač ovu osobinu iskoristi u rekurzivnom pozivu, te počne rekurzivno da preuzima entitete, dobija se XML poruka koja vrši "eksploziju zauzeća memorije", po čemu je ovaj napad i dobio naziv. Kao rezultat dešava se XDoS [6],[7],[8]. Po [9] ova vrsta napada je jedna od najčešćih.

Injection napadi – *Injection* napadi, odnosno napadi takozvanog ubrizgavanja malicioznog koda, dešavaju se kada softver ne vrši adekvatnu validaciju ulaznih podataka. Napadač u takvim slučajevima može da kreira maliciozan ulaz koji dalje može da izazove izvršavanje malicioznog koda od strane web servisa. Više različitih napada može se svrstati u ovu vrstu napada, a najvažniji su: *SQL Injection* i *XPath Injection*.

SQL Injection napadi u SOA podrazumijevaju ubacivanje SQL fragmenata u XML poruke u cilju izmjene semantike originalnog SQL upita. Izvršavanjem ovih napada, osim dobijanja podataka iz baze podataka koju koristi ranjiva web aplikacija, moguće je ostvariti i druge ciljeve, kao što su: izmjena podataka u bazi, narušavanje strukture baze podataka ili njeno brisanje. Jasno je da je za ostvarivanje navedenih ciljeva neophodno da korisnički nalog koji aplikacija koristi za pristupa bazi podataka ima za to odgovarajuće privilegije.

XPath Injection je analogija *SQL Injection* napadu kada je u pitanju rad sa XML bazama podataka. Napadom na nezaštićen servis napadač može da dođe do informacija o XML bazi jednostavnim ubacivanjem malicioznog koda u XPath naredbe.

Curenje informacija (*eng. Information Leakage*) – Programeri često generišu veoma obimne izvještaje o greškama koje jesu od velike pomoći i programerima i administratorima sistema, jer im daju informacije neophodne za otklanjanje tih grešaka. Međutim, te iste poruke mogu potencijalnom napadaču obezbijediti dodatne i veoma značajne informacije o informacionom okruženju web servisa. Nadalje, kroz WSDL opis servisa, svako može da dobije detaljan opis servisa tj. da sazna sve operacije koje servis nudi, koliko parametara kojeg tipa koja metoda prihvata kao argumente, te kojeg tipa podataka je rezultat. Na ovaj način, potencijalni napadač može detaljno da se upozna sa sistemom, što kod ranijih distribuiranih tehnologija nije bilo moguće. Ova činjenica predstavlja značajan sigurnosni problem jer napadaču daje punu informaciju o tome kako da korektno pripremi ulazne podatke za slanje zahtjeva web servisu, čime se značajno olakšava izvođenje *Injection* ili DoS i XDos napada.

Napadač može da mijenja vrijednosti parametara (*eng. Parameter Tampering*) ubacujući specijalne karaktere ili neočekivan sadržaj koji može da dovede do DoS ili neovlaštenog pristupa osjetljivim podacima [7].

Isto tako, servis može biti isprojektovan tako da je samo dio njegovih operacija namijenjen za eksternu upotrebu odnosno za pozivanje preko Interneta. Ako su u WSDL opisu objavljene i sve druge funkcije namijenjene za upotrebu iz privatne mreže vlasnika, napadač ih svejedno može pozivati [6].

Nesigurne komunikacije – Nesigurnost komunikacionih kanala oduvijek je predstavljala veoma značajan problem kod distribuiranih sistema. Međutim, neki od distribuiranih sistema koji su prethodili SOA sistemima omogućavali su ograničavanje pristupa sistemu. Na ovaj način relativno lako je onemogućiti napade. Sa druge strane, SOA uglavnom podrazumijeva komunikaciju kroz otvorenu javnu mrežu što je čini posebno osjetljivom na problematiku nesigurnih komunikacija. Ako komunikacija nije dodatno zaštićena primjenom kriptografskih algoritama i tehnika, napadač koji osluškuje komunikacione kanale može da izvrši presretanje i izmjenu poruke u prenosu, njenu modifikaciju, retransmisiju i slično. Napadač može da samo prisluškuje nezaštićen kanal (*eng. passive wiretapping*) bez vršenja ikakvih modifikacija presretnutih poruka što predstavlja prijetnju tajnosti. Pored toga, napadač može da izvrši presretanje i modifikaciju poruka (*eng. active wiretapping*) što je prijetnja autentičnosti poruka. Još jedna prijetnja koja se pojavljuje kod nesigurnih komunikacija je i mogućnost da se u nekoj od tačaka rutiranja izvrši zlonamjerno rutiranje poruke na pogrešnu adresu (takozvani *man-in-the-middle* napad) na kojoj se poruke mogu neovlašteno prihvatati, obrađivati i dalje slati na tačnu adresu. Sistemi se često štite upotrebom „standardnih" mrežnih *firewall* uređaja, ali kako ovi uređaji provjeravaju samo TCP, IP i HTTP zaglavlja, a ne i samu SOAP poruku, u SOA oni ne mogu da pruže odgovarajući nivo zaštite [6].

Reply Attack – Zaštitom SOAP poruka od neovlaštenih promjena obezbjeđuje se sigurno prenošenje poruka između legitimnog klijenta i web servisa. Međutim, ovim vidom zaštite nije onemogućena višestruka retransmisija legitimih poruka. Na ovaj način napadač legitimnim sadržajem može da izazove zagušenje servisa i tako dovesti do pojave DoS-a i/ili višestrukog neželjenog ponavljanja zahtjevane operacije web servisa.

Schema Poisoning – XML shema obezbjeđuje informacije XML parserima o očekivanom formatu XML dokumenta, kako bi oni mogli korektno da interpretiraju pristigle XML dokumente. Napadač može da pokuša da izvrši izmjenu legitimne XML sheme mijenjajući je sličnom, ali ipak izmijenjenom. Na ovaj način dešava se DoS jer će sve legitimne poruke upućene servisu biti odbačene, zato što neće zadovoljiti format definisan u izmijenjenoj XML shemi [7].

SOAP prilozi (*eng. SOAP attachments*) – Baš kao i standardne e-mail poruke, SOAP poruke mogu da sadrže priloge. Ukoliko u servis nije ugrađena kontrola SOAP priloga, napadač uz SOAP poruku može da priloži veoma velike dokumente ili dokumente koje je veoma teško obraditi što može da dovede do zagušenja komunikacionih kanala ili samog servisa. Pored toga, prilog može da sadrži i viruse koji, ukoliko se blagovremeno ne otkriju, mogu da nanesu veliku štetu.

Nekompletna ili nesigurna konfiguracija – Web servisi se gotovo po pravilu izvršavaju na serverima koji su izloženi i javno dostupni. Takva pozicija ih čini posebno osjetljivim zbog nekompletnog ili pogrešnog konfigurisanja aplikativnih servera na kojima se nalaze. Ako ovi serveri nisu pravilno konfigurisani ili ako, na primjer, nemaju instalirane najnovije sigurnosne zakrpe, napadač takve osjetljivosti servera može iskoristiti i za napad na sam servis.

Nedovoljno sigurna autentifikacija – S obzirom da su po definiciji web servisi javno objavljeni i dostupni, nedovoljno sigurno autentifikovanje korisnika može da predstavlja značajan problem. Ako se prilikom autentifikacije ne koriste sigurne veze ili jednokratni autentifikacioni kodovi, tada postoji opasnost da napadač koji prisluškuje komunikacioni kanal dođe do legitimnih pristupnih podataka koje onda može da zloupotrijebi za izvršavanje napada na sistem.

4. NAJVAŽNIJE TEHNIKE ZA OBEZBJEĐIVANJE SIGURNOSTI KOD SOA

U prethodnim sekcijama ukratko je opisano više sigurnosnih aspekata i napada kod SOA koji mogu značajno uticati na sigurnost servisno orijentisanih aplikacija. Međutim, imajući u vidu sve pobrojane aspekte i napade, korištenjem odgovarajućih sigurnosnih tehnika moguće je razviti sigurne servisno orijentisane aplikacije. U nastavku rada daje se kratak opis tehnika kojima se može spriječiti zloupotreba prethodno pobrojanih sigurnosnih aspekata.

XDoS – Kako su web servisi izloženi na javnoj mreži te im de-facto može pristupiti svako, izuzetno je važno vršiti primarno preventivno filtriranje XML poruka. Zapravo, gotovo jedini način zaštite od XDoS napada jeste puna gramatička validacija poruka na aplikativnom nivou, prije njihovog prosljeđivanja servisu na obradu [6]. Na ovaj način web servis se može zaštititi od DoS napada koji nastaju pokušajem zagušenja komunikacionih kanala, slanjem velikog broja XML poruka. Pored sprječavanja zagušenja, *XML firewall* može da se iskoristi i za kontrolu propusnih opsega ka pojedinim klijentima. Nadalje, *XML firewall*-ima, *XML gateway*-ima i XML parserima može se ograničiti i veličina XML poruka, maksimalna dužina XML elementa, maksimalan broj XML elemenata, maksimalan broj ugniježdenih XML elemenata, broj XML atributa, broj XML čvorova, lista prihvatljivih DTD-ova, dužina naziva XML čvora, te referenciranje eksternih entiteta. Na ovaj način, već u procesu prijema XML poruke, te bez implementacije dodatnih sigurnosnih mehanizama na nivou samog web servisa, može se izvršiti onemogućavanje značajnog broja XDoS napada. Ponekad se koriste i shema validatori, kao metod za validaciju XML ulaza, ali validacija je u ovom slučaju od male pomoći jer je prethodno potrebno da parser parsira XML poruku.

Injection napadi – *Injection* napadi svih tipova u značajnom obimu mogu da se osujete obaveznom i dobrom validacijom svih ulaznih podataka, prije njihove obrade i generisanja odgovarajućeg rezultata. Programer nikada ne smije da pretpostavi da će svi klijenti uvijek generisati validan ulaz za web servis, a na osnovu objavljene WSDL specifikacije.

Primjera radi, da bi *SQL Injection* napad u SOA uspio, sljedeće dvije pretpostavke moraju biti zadovoljene:

- podaci koje servis primi direktno se ubacuju u SQL naredbu i
- SQL narebe servis izvršava sa dovoljno privilegija za izvršavanje maliciozne SQL naredbe (napada).

Očigledno, da bi se ovakav vid napada spriječio, veoma je važno obraditi sve ulazne podatke prije nego što se isti uključe u sastav SQL naredbi, te ograničiti prava i privilegije korisničkog naloga koji web servis koristi za pristup bazi podataka. Kod formiranja SQL upita uvijek je potrebno koristiti parametrizovane upite, a nikada kreirati upite konkatenacijom stringova.

Analogno, *XPath injection* napad može lako da se blokira parsiranjem ulaznih podataka i osiguravanjem da podaci koji se prosljeđuju XPath naredbi u sebi ne sadrže XPath.

Curenje informacija – Programeri i administratori sistema mogu u značajnoj mjeri da utiču na curenje informacija. Prije svega, potrebno je ograničiti obim opisa grešaka te opise zapisivati u interne log datoteke. Nadalje, premda je WSDL opis servisa dio SOA standarda, WSDL opis može da bude u cjelosti uklonjen ili mu se može ograničiti pristup uz obaveznu prethodnu autentifikaciju klijenta koji želi da ga pročita. Takođe, potrebno je ograničiti sadržaj WSDL opisa na samo onaj dio web servisa koji je predviđen da se koristi sa pristupne mreže na kojoj je opis objavljen [7].

Nesigurne komunikacije – Za komunikaciju web servisa sa klijentima potrebno je koristiti kriptovane komunikacione kanale, tj. koristiti najnovije dostupne verzije SSL-a (*eng. Secure Socket Layer*) i TLS-a (*eng. Transport Layer Security*) ili enkripciju na nivou poruke. Obavezna dvostrana autentifikacija prije uspostavljanja komunikacije značajno povećava nivo povjerenja i zaštite, te značajno umanjuje mogućnost neovlaštenog prisluškivanja. Kako web servisi dozvoljavaju da SOAP poruke budu dostavljene proizvoljnom putanjom tj. dozvoljava da poruke budu rutirane kroz više tačaka izvjesno je da će na jednoj ili više tačaka rutiranja doći do terminacije SSL-a odnosno TLS-a što znači da u nekim dijelovima putanje SOAP poruka neće biti zaštićena. Jedna od najvažnijih specifikacija koja adresira problem sigurnosti u SOA jeste *WS-Security* [10]. Unutar specifikacije definisani su sigurnosni mehanizmi kao što su *XML Encryption*, *XML Signature* i *WS-SecurityPolicy*. *XML Signature* omogućava digitalno potpisivanje XML fragmenata čime se dokazuje integritet i autentičnost fragmenta. Rezultat potpisivanja smješta se u sigurnosno zaglavlje poruke (*eng. security header*) poruke, inače definisano u *WS-Security*. *XML Encryption* omogućava kriptovanje XML fragmenata čime se obezbjeđuje povjerljivost podataka. Pored toga, *WS-Security* definiše i sigurnosne tokene pogodne za prenos digitalnog identiteta kao što je X.509 certifikat ili `UsernameToken` [10]. Na kraju, *WS-SecurityPolicy* daje XML sintaksu za definisanje svih prethodno nabrojanih sigurnosnih mehanizama. U SOA realizaciji neophodno je koristiti sve ove mehanizme.

Kako je ranije navedeno, primjena tradicionalnih *firewall* uređaja za blokiranje neovlaštene komunikacije kod SOA nema efekta. U tu svrhu potrebno je koristiti *XML firewall*-e koji vrše provjeru i sadržaja komunikacije. Naime, ovakvi *firewall*-i imaju mogućnost pregledanja i analize dolaznih SOAP zahtjeva, te mogu da vrše određene akcije shodno utvrđenom sadržaju poruka. Naravno i primjena *XML firewall*-a ima svoja ograničenja, jer ni oni ne mogu da analiziraju poruke koje su kriptovane. U smislu svega prethodno navedenog, jasno je da je jedino potpuno i pravo rješenje dobro i sigurno isprojektovan i implementiran servis.

Reply Attack – Enkripcijom i korištenjem digitalnih potpisa može se osigurati zaštita web servisa od neovlaštenog prisluškivanja ili izmjene legitimnih poruka. Međutim, na ovaj način ne može da se riješi problem *reply* napada. Jedna od

tehnika kojom se može osujetiti ova vrsta napada jeste korištenje vremenskog pečata (eng. *timestamp*). Druga tehnika podrazumijeva korištenje jednokratnih identifikacionih kodova.

Već u drugoj generaciji SOA-e u okviru WS-ekstenzija kreirana je već spomenuta *WS-Security* specifikacija koja podržava rad sa vremenskim pečatima i *WS-SecurityPolicy* koja se može iskoristiti za definisanje politike u kojoj se korisnici servisa obavezuju da dostavljaju samo poruke "ovjerene" vremenskim pečatom. Ukoliko dođe do retransmisije poruke takvu poruku je lako detektovati zbog isteklog vremenskog pečata. Naravno, ključni aspekt kod ove tehnike jeste pažljiv izbor veličine vremenskog prozora u okviru kojeg poslana poruka mora da stigne do web servisa da bi bila prihvaćena. Vremenski interval mora biti dovoljno kratak da napadač nema dovoljno vremena da otkrije, presretne, dekriptuje i izvrši retransmisiju legitimne poruke, a opet dovoljno dug da bi omogućio sigurnu dostavu legitimne poruke od pošiljaoca do web servisa i u slučaju otežane komunikacije. Potrebno je izbjeći situaciju da legitimna poruka bude blokirana.

Alternativno, može se koristiti tehnika verifikacije validnosti poruke korištenjem jednokratnih identifikacionih kodova bez ili u kombinaciji sa vremenskim pečatom. Tehnika se zasniva na automatizovanom generisanju jednokratnog identifikacionog koda na osnovu vremena slanja poruke i predefinisane šifre klijenta. U ovom slučaju, klijent prilikom formiranja i slanja SOAP poruke na osnovu vremena kreiranja poruke i svoje šifre generiše jedinstveni identifikacioni kod koji šalje u sastavu same SOAP poruke ili kao prilog poruci. Po prijemu ovakve poruke, web servis samostalno izračunava taj isti identifikacioni kod te ga upoređuje sa identifikacionim kodom koji je stigao u okviru poruke i prihvata samo one poruke kod kojih se identifikacioni kodovi podudaraju. I kod ove tehnike potrebno je korektno izabrati vremenski prozor unutar kojeg će algoritam generisati isti identifikacioni kod kako ne bi došlo do odbacivanja validnih poruka.

SOAP prilozi (*eng. SOAP attachments*) – U suštini postoje dvije osnovne tehnike zaštite od napada zlonamjernim SOAP prilozima. Prva i najočiglednija tehnika jeste potpuno blokiranje priloga, tj. implementacija servisa na način koji ne zahtijeva njihovo korištenje. Ukoliko je servis takav da je nemoguće izbjeći korištenje SOAP priloga, tada se obavezno mora vršiti dvojako filtriranje priloga i to prvo filtriranje bazirano na MIME (*eng. Multipurpose Internet Mail Extensions*) tipu priloga, a onda i prosljeđivanje priloga kroz antivirusni skener.

Nekompletna ili nesigurna konfiguracija – Prilikom objavljivanja servisa, neophodno je prethodno obratiti pažnju na sigurnost sistema na kojem će se servis izvršavati. Ako u sistemu postoje druge aplikacije koje su namijenjene za upotrebu sa intraneta, poželjno je da se takve aplikacije razdvoje od aplikacija koje su predviđene za pristup sa Interneta, a samim tim i web servisa. Dobra praksa je da se interne aplikacije izvršavaju na odvojenim serverima od aplikacija koje su predviđene i za eksternu upotrebu. Pored toga, neophodno je redovno vršiti sigurnosne provjere sistema, te redovno primjenjivati sve sigurnosne zakrpe.

Nedovoljno sigurna autentifikacija – S obzirom da su po definiciji web servisi javno objavljeni i dostupni, nedovoljno sigurno autentifikovanje korisnika može da ugrozi sigurnost aplikacije. Ako se za autentifikaciju korisnika koristi samo korisničko ime i lozinka, tada se taj proces mora vršiti minimalno kroz HTTPS.

5. ZAKLJUČAK

Ključ za kreiranje sigurnih aplikacija i aplikativnih arhitektura jeste razumijevanje sigurnosnog profila odnosno mogućih osjetljivosti i napada na sistem. Svaki put kada se pojavi nova arhitektura neizbježno dolazi i do promjene sigurnosnog profila. SOA nije izuzetak.

U ovom radu, koliko je obim rada dozvoljavao, opisani su najvažniji sigurnosni aspekti SOA koji su se pojavili u SOA ili koji u SOA imaju posebne specifičnosti. Pored toga, opisano je više tehnika kojima se mogu spriječiti napadi na nabrojane osjetljivosti SOA.

Premda SOA ima niz sigurnosnih osjetljivosti koje se mogu iskoristiti za vršenje napada na SOA, prvenstveno dobrim projektovanjem i implementacijom servisa, a onda i obaveznim korištenjem *WS-Security*, *WS-SecurityPolicy*, *XML Encryption*, *XML Signature* specifikacija, shema validacijom XML poruka, korištenjem *XML Firewall*-a, validacijom ulaznih parametara, antivirusnom kontrolom dolaznih poruka, pažljivim objavljivanjem WSDL opisa i drugim nabrojanim tehnikama moguće je kreirati dovoljno sigurnu servisno orijentisanu aplikaciju.

6. LITERATURA

[1] Saša Marić, "Projektovanje servisno orijentisane aplikacije", diplomski rad, 2007.

[2] Thomas Erl, "Service-Oriented Architecture - Concepts, Technology and Design", 3rd edition, Prentice Hall, 2005.

[3] Charles P. Pfleeger, "Security in Computing", 4th Edition, Prentice Hall, oktobar 2006.

[4] Ramarao Kanneganti, Prasad Chodavarapu, "SOA Security", Manning Publications Co., 2008.

[5] Gunnar Peterson, "Service Oriented Security Architecture", Information Security Bulletin, CHI Publishing, Vol. 10, pp. 325-330, Nov. 2005.

[6] Nils Gruschka, Norbert Luttenberger, "Protecting Web Services from DoS Attacks by SOAP Message Validation", Proceedings of the IFIP TC11 21 International Information Security Conference, 2006.

[7] Pete Lindstrom, "Attacking and Defending Web Services", A Spire Research Report, januar 2004.

[8] http://www.networkworld.com/news/2009/031209-soa-security-the.html, posjećivano februara 2011. godine

[9] "Symantec Global Internet Security Threat Report: Trends for 2009" (PDF), Symantec Corp., Vol XV, april 2010.

[10] Meiko Jensen, Nils Gruschka, Ralph Herkenhöner, "A survey of attacks on web services", Computer Science - Research and Development,Vol. 24, No 4, pp185-197, 2009.