Centar Informacijske Sigurnosti

Ispitivanje sigurnosti web servisa

veljača 2012.

CC BY NC SA

CIS-DOC-2012-02-040

Upozorenje

Podaci, informacije, tvrdnje i stavovi navedeni u ovom dokumentu nastali su dobrom namjerom i dobrom voljom te profesionalnim radom CIS-ovih stručnjaka, a temelje se na njihovom znanju i petnaestak godina iskustva u radu u informacijskoj sigurnosti. Namjera je da budu točni, precizni, aktualni, potpuni i nepristrani.

Ipak, oni su dani samo kao izvor informacija i CIS ne snosi nikakvu izravnu ili posrednu odgovornost za bilo kakve posljedice nastale korištenjem podataka iz ovog dokumenta.

Ukoliko primijetite bilo kakve netočnosti, krive podatke ili pogreške u ovom dokumentu, ili imate potrebu komentirati sadržaj molimo Vas da to javite elektroničkom poštom na adresu info@CIS.hr.

O CIS-u

CIS izrađuje pregledne dokumente (eng. white paper) na teme iz područja informacijske sigurnosti koji će biti korisni zainteresiranoj javnosti, a u svrhu **podizanje njezine svijesti o informacijskoj sigurnosti** i **sposobnosti za čuvanje i zaštitu informacija i informacijskih sustava**. Pored toga, CIS razvija i održava mrežni portal www.CIS.hr kao referalnu točku za informacijsku sigurnost za cjelokupnu javnost; izrađuje obrazovne materijale namijenjene javnosti; organizira događaje za podizanje svijesti o informacijskoj sigurnosti u javnosti i pojedinim skupinama te djeluje u suradnji sa svim medijima.

CIS **okuplja mlade** zainteresirane za informacijsku sigurnost i radi na njihovom pravilnom odgoju i obrazovanju u području informacijske sigurnosti te pripremu za **profesionalno bavljenje informacijskom sigurnošću**.

Centar informacijske sigurnosti [CIS] nastao je 2010. godine na poticaj Laboratorija za sustave i signale[LSS] Zavoda za elektroničke sustave i obradbu informacija Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu, a kao posljedica 15togodišnjeg rada na istraživanju, razvoju i primjeni informacijske sigurnosti. LSS je među ostalim potaknuo osnivanje CARNetovog CERTa i sudjelovao u izradi Nacionalnog programa informacijske sigurnosti RH.

Smisao CISa je da bude **referentno mjesto za informacijsku sigurnost** za javnost, informatičare i posebno za mlade te da sustavno podiže njihovu svijest i sposobnosti u području informacijske sigurnosti.

Rad CISa podržava Ministarstvo znanosti, obrazovanja i sporta Republike Hrvatske, a omogućuju sponzori.

Prava korištenja

Ovaj dokument smijete:

- Dijeliti - umnožavati, distribuirati i priopćavati javnosti,
- Remiksirati - prerađivati djelo

pod slijedećim uvjetima:

- Imenovanje - Morate priznati i označiti autorstvo djela na način da bude nedvojbeno da mu je autor Laboratorij za sustave i signale, Fakulteta elektrotehnike i računarstva, Sveučilišta u Zagrebu.
  To morate napraviti na način koji ne sugerira da Vi ili Vaše korištenje njegova djela imate izravnu podršku LSSa.
- Nekomercijalno - Ovo djelo ne smijete naplaćivati ili na bilo koji način koristiti u komercijalne svrhe.
- Dijeli pod istim uvjetima - Ako ovo djelo izmijenite, preoblikujete ili koristeći ga stvarate novo djelo, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj i pri tome morate označiti izvorno autorstvo Laboratorija za sustave i signale, Fakulteta elektrotehnike i računarstva, Sveučilišta u Zagrebu.

Detalji licence dostupni su na: http://creativecommons.org/licenses/by-nc-sa/3.0/hr/legalcode

Sadržaj

1. Uvod

Web servisi su proizvod novog načina projektiranja informacijskih sustava. Širenje ideje o raspodijeljenoj obradi podataka i dijeljenju funkcionalnosti s drugim organizacijama uzrokovalo je ubrzani razvoj samog weba. Naime, organizacije više nisu usmjerene isključivo na razvoj vlastitih aplikacija kako bi promovirale ili unaprijedile vlastito poslovanje. Novom metodologijom razvoja, one određeni dio vlastitog poslovanja otvaraju zajednici i čine ga dostupnim široj javnosti. Primjerice, Internet bankarstvo je do nedugo bilo moguće obavljati isključivo putem web aplikacije koja je pod domenom banke. Danas banke nude korisnicima pristup svojim web servisima kojima je moguće ispuniti iste funkcionalnosti kao i s web aplikacijom. Time korisnik više nije ograničen na korištenje jedne web aplikacije kako bi obavio usluge Internet bankarstva. Osim toga, druge organizacije mogu putem web servisa banke nuditi veću razinu usluge u svojem poslovanju (npr. kupovina putem Interneta). Izlaganjem vlastitih funkcionalnosti u obliku web servisa, banka je osigurala veću prilagodljivost vlastite usluge i time povećala svoj ugled. Istovremeno je omogućila drugim organizacijama da povećaju vlastitu razinu usluge. Obje strane imaju veliku korist od ovakve suradnje, a korisniku se pruža bolje iskustvo prilikom korištenja usluga.

Obzirom na ovakve obostrane pogodnosti, web servisi su postali obavezni dio u poslovanju gotovo svake organizacije. Ipak, dijeljenje funkcionalnosti znači i dijeljenje njezinih ranjivosti i nedostataka. Točnije, sigurnosne ranjivosti jednog servisa sada imaju utjecaj i na poslovanje organizacija koje taj servis koriste. Porast uporabe web servisa u svakodnevnom poslovanju povećava potrebu za konkretnim metodologijama ispitivanja sigurnosti istih. Trenutno ne postoji jedinstvena metodologija ispitivanja sigurnosti web servisa koja obuhvaća sve moguće scenarije. No, postoje određeni prijedlozi metodologija i preporučenih koraka koji olakšavaju postupak ispitivanja servisa. Neke od tih metodologija su slične onima za ispitivanje web aplikacija. Iako su web servisi načelno slični web aplikacijama, postoje velike razlike u njihovoj ideologiji i samom načinu ispitivanja. Zbog sve većeg broja novih tehnologija i standarda koji se koriste za oblikovanje servisa, njihovo sigurnosno ispitivanje postaje znatno složenije od web aplikacija.

U ovom dokumentu se opisuju neki od najpoznatijih standarda i tehnologija za ostvarivanje i osiguravanje web servisa. Poglavlje 2 daje općeniti opis postojećih metodologija razvoja web servisa i njihov model, dok se u poglavlju 3 razmatraju tehnologije za osiguravanje web servisa. Ovisno o korištenoj metodi razvoja web servisa i korištenim tehnologijama i standardima, način ispitivanja sigurnosti web servisa se može razlikovati. U poglavlju 4 se opisuju neke od metoda ispitivanja kao i alati koji mogu ubrzati postupak provjere.

2. Web servisi

Računarstvo zasnovano na uslugama (engl. *SOA – Service Oriented Architecture*) predstavlja novi pokret u računarstvu koji se zasniva na izradi javno dostupnih servisa pomoću kojih se obavlja određena funkcionalnost. Točnije, pojedine funkcionalnosti se više ne obavljaju putem uobičajenih programskih alata već se ukupna funkcionalnost sustava gradi manjim gradivnim blokovima koji se nazivaju servisi. Ovisno o primjeni, servisi mogu biti jednostavniji ili složeniji. Osim toga, sustavi mogu kombinirati više jednostavnih ili složenih servisa kako bi korisnicima pružili potpuniju uslugu. Osnovna pretpostavka ovakvih sustava je dostupnost servisa putem globalne mreže - Interneta. Iz tog razloga koristi se pojam web servisi kako bi se naglasila ovisnost o Internetu.

Računarstvo zasnovano na uslugama postoji već neko vrijeme i koristi se u različite svrhe. Jedno od konzistentnih tumačenja je pristup izradi informacijskih sustava koji označava podjelu ukupnih zahtjeva na niz manjih. Ovakvo sitnozrnato ostvarivanje funkcionalnosti omogućuje ponovnu uporabu pojedinih dijelova i olakšava njihovo ostvarivanje. Naime, rastavljanjem složenih zahtjeva u niz jednostavnijih smanjuju se moguće količine grešaka i ubrzava vrijeme razvoja. Ovaj pristup nadilazi granice tehnologije i razvoja informacijskih sustava kao općenita smjernica za razvoj programske potpore. Ono po čemu se princip računarstva zasnovanog na uslugama razlikuje od rastavljanja zahtjeva je u načinu na koji se postiže odvajanje.

Razlike između računarstva zasnovanog na uslugama i rastavljanja zahtjeva se mogu opisati usporedbom web servisa sa svakodnevnim danom u gradu. Svaki grad se sastoji od određenog broja organizacija koje obavljaju svoje djelatnosti i nude određene usluge. Svaku uslugu može koristiti više klijenata. Na primjer, kemijske čistionice nude svim svojim korisnicima usluge čišćenja. Čistionicu može koristiti više klijenata od jednom, ali nisu nužno ovisni jedni o drugom. Svi pružatelji usluga čine jednu zajednicu, ali nisu nužno povezani jedno s drugim te nisu međusobno ovisni. Web servisi čine gotove funkcionalnosti koje se mogu pozivati iz različitih alata i okruženja. Svaki servis mora imati dobro opisano pristupno. Kako bi se mogli koristiti potrebno je preuzeti navedeni opis pristupnog sučelja. Većina sučelja se mogu dohvatiti iz javnih registra koji imaju popise poznatih web servisa. Kada se dohvati opisnik sučelja korisnik ili programski alat stvara zahtjeve prema web servisu na način koji je propisan opisnikom. U opisniku se nalazi pristupna adresa, mrežna vrata, opis ulaznih parametara i druge upute koje su potrebne za korištenje servisa. Konkretna izvedba opisnika sučelja, registra servisa i način interakcije s korisnikom ili alatom ovisi o arhitekturi i tehnologiji koja se koristi prilikom izrade web servisa. Trenutno najrašireniје arhitekture za ostvarivanje web servisa su REST (engl. *Representational state transfer*) i WS-Services koji se opisuju u nastavku poglavlja. Ipak, postoje općenite smjernice i načela koje valja pratiti prilikom izrade web usluga. Neke od njih su u nastavku(više detalja se može pronaći u dodatnoj literaturi pod **Error! Reference source not found.**):

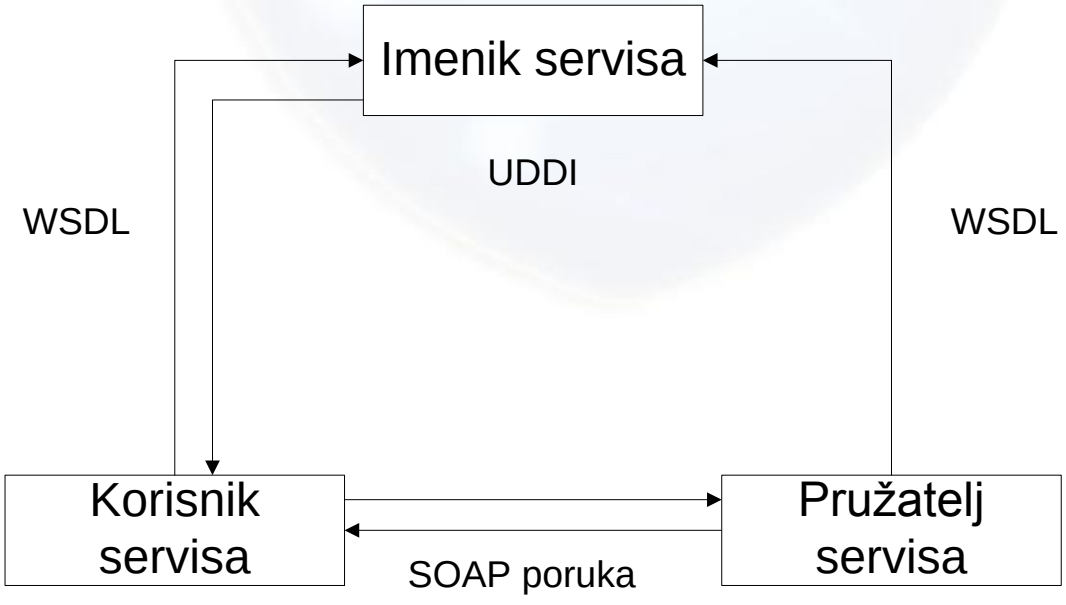
- **Slaba povezanost** – označava nisku ovisnost između web servisa. Slaba povezanost je jedna od najvažnijih načela izgradnje web servisa jer izravno ostvaruje ideju skupa nezavisnih javno dostupnih funkcionalnosti. Postoje razni tipovi povezanosti kao što su prostorna, vremenska, strukturna i druge. Na primjer, kod prostorne ovisnosti uspješnost izvođenja usluge ili sustava ovisi o geografskom položaju. Vremenska ovisnost nastaje kada se web servis može naći u stanju u kojem ne može dati odgovor. Strukturna ovisnost nastaje kada web servis može interpretirati samo podatke koji su strogo strukturirani po nekom predlošku.
- **Krupna zrnatost** – podrazumijeva manji broj usluga složenije funkcionalnosti. Zrnatost opisuje mjeru složenosti funkcije koju usluga ostvaruje. Krupna zrnatost ima određene prednosti zbog manjeg vremena odziva i složenosti orkestracije velikog broja manjih usluga. Naime, kada se koristi više jednostavnijih usluga za ostvarenje jedne složene uvode se ovisnosti o vanjskim sustavima koje nisu uvijek prihvatljive. Dodatno, povećava se broj mogućih točaka ispada. Web servisi se često oblikuju po uzoru na usluge stvarnog svijeta što prirodno povećava zrnatost rješenja. Odabir zrnatosti je složen postupak, a više informacija se može pronaći u dodatnoj literaturi pod [2].
- **Otvoreni standardi** – kako su web servisi namijenjeni za uporabu u svim okruženjima, uporaba otvorenih standarda je ključna kako bi se zaobišla platformska ovisnost. Uporaba otvorenih standarda pogoduje i načelu slabe povezanosti budući da predstavlja najbliži korak

prema platformskoj neovisnosti web servisa. Postoje razni standardi za opis pristupnog sučelja i tehnologije ostvarenja web servisa. Neki od njih su detaljnije opisani u nastavku poglavlja.

- **Prilagodljivost** – mogućnost izmjene vlastitih radnih postavki tijekom izvođenja. Ovime se želi povećati učinkovitost servisa bez izmjena samog sustava ili okoline. Dodatno, olakšava se postupak održavanja obzirom da većinu postavki servis samostalno mijenja. Prilagodljivost obuhvaća velik skup aktivnosti koje web servisi mogu u pozadini raditi kako bi osigurali bolji odziv ili kvalitetnije rezultate. Neki servisi nude više pristupnih sučelja ovisno o željenoj kvaliteti usluge. Različite razine kvalitete nisu nužno dostupne svim korisnicima, nego samo manjem podskupu. Primjerice, neki servisi nude bolji odziv registriranim korisnicima ili korisnicima koji su platili pristup višoj razini kvalitete.

2.1. WSDL, SOAP i UDDI

Zahvaljujući svojoj popularnosti i praktičnosti, web servisi su doživjeli nagli razvoj u posljednjih nekoliko godina. Nastale su nove tehnologije koje olakšavaju razvoj i interakciju sa servisima. Jedan od prvih standarda za opis web servisa bio je WSDL (engl. *Web Services Description Language*). Riječ je o jezik zasnovanom na XML (engl. *Extensible Markup Language*)-u, koji se koristi za opisivanje sučelja web servisa. WSDL definicijom web servisa daje se opis svih potrebnih ulaznih i izlaznih parametara te pristupna adresa servisa. Takav opis je pogodan jer je razumljiv ljudima i računalima. Prilikom složene orkestracije više web servisa (opisano u poglavlju 2.3.2) često se upiti samostalno formiraju uporabom WSDL definicije pristupnog sučelja. Prije nego što korisnik može preuzeti WSDL opis web servisa, pružatelj ga mora objaviti. U tu svrhu se obično koristi javni imenik web servisa koji sadrži pristupna sučelja. Najkorišteniji standard za ostvarenje imenika servisa je UDDI (engl. *Universal Description, Discovery and Integration*). UDDI je otvoren svim zainteresiranim stranama, a podržava ga organizacija OASIS (engl. *Organization for the Advancement of Structured Information Standards*). Nakon što korisnik učita WSDL opis potrebno je pristupiti željenom web servisu. Kao protokol za razmjenu poruka između korisnika i servisa koristi se SOAP (engl. *Simple Object Access Protocol*). Ovaj protokol je razvijen isključivo za komunikaciju s web servisima, a od 2003. godine postaje dio W3C preporuka. Slika 1. prikazuje proces korištenja web servisa od objave WSDL opisa u javni imenik do razmjene poruka između korisnika i pružatelja servisa. Više informacija o ovim standardima može se pronaći u dodatnoj literaturi pod [1] i [3].



Slika 1. Komunikacija s web uslugom
Izvor: Centar Informacijske Sigurnosti

2.2. REST usluge

REST (engl. *Representational State Transfer*) predstavlja teorijski model programske arhitekture za ostvarivanje raspodijeljenih sustava, a opisao ga je Roy Fielding u svojoj doktorskoj disertaciji [10]. Nastao je iz WWW (engl. *World Wide Web*) tehnologije uvođenjem određenih ograničenja. Ova ograničenja čine osnovne principe REST modela koji određuju kako se resursi na globalnoj mreži Internet mogu koristiti. Motivacija za uvođenje ovih ograničenja je stvaranje konačnog sustava koji koristi sve pogodnosti arhitekture weba kako bi ostvario bolje funkcioniranje novog sustava. Budući da web servisi često znaju implementirati samo dio teorijskih načela REST modela, koristi se naziv RESTful web servisi. Time se naglašava da se koriste neka REST načela, ali ne sva. U nastavku se nalazi opis ključnih načela REST modela, a više informacija može se pronaći u dodatnoj literaturi pod [1], [3], [5], [9] i [10].

- **Svako sredstvo ima jedinstveni identifikator** – osnovni gradivni elementi REST modela su sredstva. Svaki web servis korisnicima nudi pristup konačnom broju sredstava. Na primjer, popularni web servisi kao što je Amazon API omogućuju dohvaćanje naslova knjiga, korisničkih kritika i druge informacije. U kontekstu ovog servisa, knjige i kritike predstavljaju sredstva koja web servis nudi korisniku. Kao jedinstveni identifikator sredstva odabran je URI (engl. *Uniform Resource Identifier*). Time sredstvo ima jedinstveni identifikator unutar globalne mreže Internet.
- **Međusobno povezivanje sredstava** – kako sredstva često mogu imati prirodnu vezu jedno s drugim potrebno ih je povezati. Na primjer, kritike i knjige kojima pripadaju su međusobno povezane semantikom. Točnije, kritike nemaju smisla bez knjige na koju se odnose, dok se knjige mogu detaljnije opisati kritikama. Iz tog razloga se prilikom davanja odgovora na zahtjev za sredstvom često daje skup dodatnih poveznica koje identificiraju druga sredstva. Time klijent ima mogućnost promjene stanja sustava prateći poveznice koje je dobio u sklopu odgovora.
- **Uporaba standardnih metoda** – svako sredstvo podržava isti skup metoda kojima se mogu pokrenuti određene operacije ili dohvatiti željeni rezultat. Skup standardnih REST metoda preuzete su iz skupa HTTP metoda. Teorijski model predviđa uporabu GET, POST, PUT i DELETE metoda. Iako je način uporabe tih metoda opisan HTTP standardom i REST teorijskim modelom, one se rijetko kada koriste dosljedno. Iz tog razloga se često koristi pojam RESTful servisi kako bi se naglasilo da neka načela nisu strogo ispunjena. Na primjer, GET metoda bi se trebala koristiti isključio za dohvaćanje sredstava. Popularni web servis Flickr ima dokumentiranu operaciju brisanja sredstva uporabom GET metode, umjesto standardne DELETE metode.
- **Sredstva s višestrukim reprezentacijama** – jedan od osnovnih problema prilikom izrade web servisa je osiguravanje ispravnog tumačenja poslanih podataka na strani klijenata. Naime, web servisi šalju samo podatke, a klijentska aplikacija ih mora prikazati. Točnije, klijentska aplikacija mora znati kako protumačiti rezultate te ih onda prikazati korisniku. Kako bi se pokrilo što više mogućih reprezentacija, moguće je HTTP zahtjevom zatražiti željeni format. Naravno, web servis mora podržavati isporuku sredstva u željenom formatu. Na primjer, kada se web servis koristi s mobilnog uređaja poželjno je dobiti odgovor u XML zapisu. U drugim uvjetima pogodnije je odgovor primiti u XML ili JSON obliku. Neovisno o odabranom formatu, klijent je uvijek zadužen za konačnu reprezentaciju resursa.
- **Komunikacija bez održavanja stanja** – REST model predlaže da se informacija o stanju servisa uvijek bude sadržana unutar samog sredstva koje se koristi ili da se pohranjuje na klijentu. Stanje se odnosi na bilo koje informacije koje utječu na rezultat odziva web servisa, a koje su nastale putem prethodnih interakcija s web servisom. Točnije, poslužitelj ne bi trebao pamtiti stanje prilikom komuniciranja s pozivateljem servisa. Jedan od razloga za to je osiguravanje razmjernog rasta. Ukoliko se pamti stanje za svakog klijentu to bi narušilo vrijeme odziva prilikom visokog opterećenja. No, postoje i druge pogodnosti ovog pristupa. Na primjer, klijent postaje neovisan o promjenama na poslužitelju jer ne ovisi o stanju prilikom komunikacije.

2.3. WS-*

Jedna od temeljnih ideja računarstva zasnovanog na web servisima je olakšati razvoj složenih sustava rastavljanjem problema na manje dijelove. Iako ovaj koncept zvuči jednostavno, izrada web servisa može vrlo brzo postati složena. Zadovoljavanje svih načela koji su predstavljeni u poglavlju 2 može biti vrlo složeno ovisno o domeni problema. Kako bi se olakšao razvoj web servisa razvijeni su razni standardi i predlošci koji se mogu primijeniti u raznim situacijama. Ovaj skup predložaka i standarda nosi naziv „WS-*“, a podijeljen je u niz potkategorija. Svaka potkategorija nudi skup otvorenih predložaka koji se mogu koristiti za izradu vlastitih servisa. Iako imaju zajednički naziv ne održava ih jedno regulacijsko tijelo ili organizacija već više nezavisnih organizacija. Samim tim se i pojedini predlošci razlikuju u kvaliteti i doprinosu. Dodatno, predlošci se u nekim slučajevima mogu međusobno nadopunjavati, poklapati, ali i nadmetati jedan s drugim. Svi standardi i predlošci zajedno čine drugu generaciju standarda koji su predstavljeni WSDL, SOAP i UDDI jezicima. Kako su web servisi postali globalno popularni, broj predložaka i potkategorija je nadrastao opseg ovog dokumenta. U nastavku se opisuju neki od najkorištenijih predložaka, a više informacija se može pronaći u dodatnoj literaturi pod [2], [3] i [4].

2.3.1. WS-Addressing

Skup predložaka WS-Addressing web servisima omogućuje usmjeravanje poruka kroz komunikacijsku infrastrukturu. Specifikacije su neovisne o transportnom protokolu koji se koristi za prijenos, a omogućuje korisniku ili usluzi proizvoljan odabir puta. WS-Addressing dodaje skup metapodataka standardnom SOAP zaglavlju kojime se upravlja kretanje poruke kroz mrežu. Mrežni sloj koji preuzima poruku je zadužen samo za prosljeđivanje poruke do posebnog otpremnog čvora koji može pročitati WS-Addressing metapodatke. Kada poruka dođe do otpremnog čvora posao mrežnog sloja je završen. Daljnje usmjeravanje obavlja otpremni čvor u skladu s metapodacima u zaglavlju.

Ukoliko se koristi WS-Addressing specifikacija potrebno je postaviti odredišnu točku servisa (engl. *EPR – Endpoint Reference*). Odredišna točka predstavlja XML strukturu koja sadrži sve željene korake za upravljanje toka podataka prilikom komunikacije. Ona uključuje točnu mrežnu adresu web servisa i sve dodatne parametre koji su potrebni za usmjeravanje poruke do tog odredišta. Dodatno, moguće je koristiti asinkrone pozive za interakciju s web servisima koristeći dodatna SOAP zaglavlja. Time se poruka prenosi preko posebne veze što pospješuje odvajanje izvorišnog sustava od vremena odziva servisa. Tablica 1. opisuje neke atribute koji su dio specifikacije. WS-Addressing su predložile organizacije Microsoft, IBM, BEA, Sun te SAP, a odobrila ga je organizacija W3C. Više o standardizaciji može se pronaći u dodatnoj literaturi pod [6].

Naziv elementa	Opis
Address	Atribut *Address* je jedini obavezni element WS-Addressing specifikacije. Koristi se za definiranje odredišne adrese web servisa.
ReferenceProperties	Predstavlja element koji može imati dodatne potkategorije kojima se opisuju određena svojstva instance web servisa.
PortType	Opisuje odabrana vrata na web servisu.
ServiceName i PortName	Atribut ServiceName označava naziv web servisa kojemu se pristupa, dok PortName naziv mrežnih vrata. Ovi elementi su ujedno i u standardnoj WSDL specifikaciji web servisa.
Policy	Element se koristi kako bi se povezao na dodatnu specifikaciju WS-Policy kojom se definiraju dodatne informacije o pravima pristupa i usmjeravanja.
ReplyTo	Atribut *ReplyTo* označava adresu otpremnog čvora koji će se dalje brinuti za preusmjeravanje do odredišta.
RelatesTo	Ukoliko se poruka sastoji od više dijelova ili se oslanja na sadržaj od prijašnje poruke ovim elementom se može izraziti ta ovisnost.
From	Označava odredišnu točku koja se koristi za slanje odgovora na upit.
MessageID	Koristi se kao jedinstveni identifikator poruke. Ovaj identifikator se može koristiti u svim do sada navedenim elementima.
Action	Element sadrži URI identifikator akcije koja će se obaviti prilikom obrade zaglavlja poruke.

Tablica 1. WS-Addressing elementi

2.3.2. WS-BPEL

Kao posljedica ideologije računarstva zasnovanog na web servisima često je potrebno kombinirati više servisa u jednoj operaciji. Upravljanje velikim brojem servisa može biti složen zadatak. Kako bi se taj proces olakšao stvoren je jezik BPEL (engl. *Business Process Execution Language*). Jezik je postao dio WS-* standarda u sklopu WS-BPEL specifikacije. Predstavlja jezik za izvođenje specifičnih akcija unutar poslovnog procesa korištenjem web servisa. BPEL procesi koriste isključivo sučelja web servisa kako bi uvozili i izvozili informacije. Kao i u svakom programskom jeziku, moguće je izvoditi naredbe grananja prema željenim kriterijima. WS-BPEL je prvenstveno zamišljen kao jezik za orkestraciju složenih web servisa koji se sastoji od većeg broja manjih. Jezici za orkestraciju se razlikuju od ostalih programskih jezika po načinu izvođenja i upravljanju toka programa. Orkestracija podrazumijeva izvođenje procesa koji razmjenjuje poruke s drugim sustavima i time ostvaruje ukupnu željenu funkcionalnost. Redoslijed razmjene poruka se definira programom, a zahvaljujući naredbama grananja moguće je donositi odluke o dodatnim akcijama ovisno o rezultatu obrade. Točnije, orkestrirano izvođenje podrazumijeva da postoji središnje mjesto upravljanja. Kod jezika WS-BPEL, ovu ulogu preuzima sam program koji koordinira pozive web servisa s drugih sustava. Iako jezik BPEL pripada novijoj generaciji tehnologija za stvaranje web servisa, podržava i starije mehanizme komunikacije kao što su WSFL, XLANG i drugi. Iz tog razloga se koristi WSDL inačice 1.1 kako bi se opisala poruka za razmjenu podataka s drugim sustavima. Kao i svi dijelovi WS-* specifikacije, WS-BPEL je programski jezik koji se opisuje jezikom XML. U nastavku se nalazi primjer jednostavnog programa u jeziku WS-BPEL, a Tablica 2. opisuje najvažnije elemente kojima se oblikuje program. Primjer programskog koda je preuzet iz

dodatne literature pod [8], a više informacija o jeziku WS-BPEL se može pronaći u dodatnoj literaturi pod [1] i [7]. Primjer u nastavku ima korijenski element *process*, kojim se definiranja cijeli WS-BPEL program. Osnovni atributi podrazumijevaju naziv programa te poveznicu na korištenu XML shemu koja se koristi za vrednovanje napisanog programa. Prvi ugniježđeni element je *partnerLinks* kojime se definiraju veze s drugim sustavima koji će se koristiti tokom izvođenja programa. Idući blok predstavlja definiciju programskih varijabli, koja ima dvije ugniježđene vrijednosti. Te vrijednosti su nove varijable *request* i *response*. U nastavku se definira niz koraka pomoću *sequence* elementa. Ovim elementom se definira pozivanje servisa i primanje odgovora.

Element	Opis
Sequence	Vršni element kojim se označava niz aktivnosti za slijedno izvođenje. Te aktivnosti se moraju ugnijezditi unutar *sequence* elementa.
Invoke	Koristi se za poziv web servisa.
Recieve	Označava čekanje primitka zahtjeva na sučelju BPEL programa.
Reply	Koristi se za slanje odgovora na zahtjev koji je zaprimljen putem *recieve* elementa.
While	Predstavlja uobičajenu kontrolnu naredbu koja omogućuje izvođenje određenog dijela naredbi dok se ne ispuni željeni uvjet.
Copy	Naredba za pridruživanje vrijednosti varijablama.
Variables	Vršni element kojim započinje blok u kojemu se definiraju varijable. Može imati jedan ili više *variable* elemenata.
Variable	Element kojim se definira konkretna varijabla. Mora se nalaziti unutar *variables* elementa.
Scope	Pojedine dijelove poslovne logike moguće je odvojiti u posebne odvojene blokove koristeći *scope* element. U tako definiranom bloku moguće je dodavati nove varijable, elemente za upravljanje greškama, komunicirati s drugim sustavima i sve ostalo što je inače omogućeno WS-BPEL programima. Elementi koji su definirani u jednom *scope* elementu nisu vidljivi izvan njega. Ima istu funkciju kao BEGIN-END blokovi ili vitičaste zagrade u standardnim programskim jezicima kao C/C++, Java i drugi.
Terminate	Element kojim se prekida WS-BPEL proces. U novijoj specifikaciji standarda, WS-BPEL 2.0 predloženo je da se ovaj element preimenuje u *exit*.
Wait	Ovim elementom se zaustavlja izvođenje procesa na određeno vrijeme nakon isteka zadanog vremena, proces se nastavlja. Moguće je zadati konkretno vrijeme u sekundama ili definirati precizan datum nastavljanja izvođenja.

Tablica 2. Elementi jezika WS-BPEL

```
<process name="HelloWorld"
targetNamespace="http://jbpm.org/examples/hello"
  xmlns:tns="http://jbpm.org/examples/hello"
  xmlns:bpel="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/">
```

```
  <partnerLinks>
    <!-- definicija odnosa s drugim servisom -->
    <partnerLink name="caller" partnerLinkType="tns:Greeter-Caller"
myRole="Greeter" />
  </partnerLinks>


  <variables>
    <!-- varijabla za pohranu upita -->
    <variable name="request" messageType="tns:nameMessage" />
    <!-- varijabla za pohranu odgovora -->
    <variable name="response" messageType="tns:greetingMessage" />
  </variables>


  <sequence name="MainSeq">

    <!-- poziv prema drugom web servisu -->
    <Receive name="ReceiveName" operation="sayHello" partnerLink="caller"
      portType="tns:Greeter" variable="request" createInstance="yes" />

    <!-- stvaranje novog upita -->
    <assign name="ComposeGreeting">
      <copy>
        <from expression="concat('Hello, ',
bpel:getVariableData('request', 'name'), '!')" />
        <to variable="response" part="greeting" />
      </copy>
    </assign>

    <!-- slanje odgovora web servisu  -->
    <reply name="SendGreeting" operation="sayHello" partnerLink="caller"
      portType="tns:Greeter" variable="response" />

  </sequence>

</process>
```

3. Sigurnost web servisa

Zahvaljujući dinamičkom razvoju tehnologije weba, računarstvo zasnovano na uslugama je doživjelo nagli razvoj. Web servisi postaju dio svakodnevice korisnika. Organizacije su većinu svojeg poslovanja ponudili korisnicima u obliku raznih web servisa. Na primjer, banke već dulje vrijeme omogućuju plaćanje putem Internet bankarstva ili putem mobilnih aplikacija. Kako web servisi nemaju jedinstveno sučelje već se mogu koristiti na svim platformama pogodni su za izdvajanje funkcionalnosti. Naime, prilikom plaćanja putem Internet bankarstva ili mobilnih aplikacija koriste se jedni te isti web servisi. Samim time, web servisi su postali očita meta zlođudnih napada na informacijske sustave. Uspješnim izvođenjem napada na web servis može se narušiti čitav niz drugih podsustava koji koriste ili su ovisni o tom servisu. Na primjer, izvođenje napada uskraćivanjem usluge (engl. *DoS – Denial of Service*) nad web servisom za provjeru stanja na računu može uzrokovati ispad svih sustava koji koriste tu uslugu. Korisnici koji putem Internet bankarstva žele uvid u svoje račune neće moći obaviti pregled. Isto tako, korisnici koji putem mobilnih uređaja žele provjeriti stanje na računu biti će zakinuti zbog ispada servisa. No, postoje i veće opasnosti ovakve infrastrukture. Ukoliko se otkrije ranjivost web servisa kojom se može postići određena korist, postoje gotovo neograničeni izvori napada. Korisnici s mobilnih uređaja, kao i oni koji koriste web aplikacije za Internet bankarstvo mogli bi iskoristiti istu ranjivost.

Kada se web servisi razmatraju sa stanovišta sigurnosti, postoje dvije očite točke izvora ranjivosti. Te točke su komunikacijski kanal putem kojeg se razmjenjuju poruke i domena posluživanja. Osiguravanje tajnosti informacija prilikom prijenosa poruka putem nesigurne mreže je jedan od osnovnih sigurnosnih problema globalne mreže Internet. Zahvaljujući razvoju tehnologija weba ovaj sigurnosni problem se riješio uvođenjem kriptografskih postupaka tajnosti i autentifikacije korisnika. Kako su kriptografija i autentifikacija korisnika široki pojmovi, ne razmatraju se u ovom dokumentu. Zainteresirani čitatelji mogu pronaći dodatne informacije u literaturi pod [11]. Drugi sigurnosni problem je domena posluživanja web servisa, a usko je vezana uz autentikaciju. Naime, potrebno je osigurati da korisnik zaista ima pravo obavljati određene operacije. Ovo se obično radi putem tablica za provjeru pristupa (engl. ACL – *Access Control List*) kojima se uparuju korisnici, resursi i dozvoljene operacije. No, ovo često nije dovoljno te se koriste dodatne akcije nadziranja. Kod kritičnih web servisa, kao što su novčane transakcije, bilježi se svaki korisnikov korak. Time bi se u slučaju napada lakše rekonstruirao zlođudni niz akcija te sankcionirao na odgovarajući način. U nastavku poglavlja se opisuju neke od poznatijih metoda zaštite web servisa. Cilj tih mehanizama je osiguravanje osnovnih sigurnosnih zahtjeva koji su navedeni u nastavku.

- **Autentikacija** (engl. *Authentication*) – potvrda autentičnosti korisnika. Odgovarajuće metode provjere autentičnosti korisnika primjenjuju se ovisno o aplikaciji i uslugama koje ih koriste.
- **Cjelovitost** (engl. *Integrity*) – jamstvo da su informacije poslane, primljene ili pohranjene u izvornom i nepromijenjenom obliku. Samo ovlaštenim osobama dopušteno je upisivanje, promjena, izmjena statusa, brisanje, stvaranje, kašnjenje ili ponavljanje podataka.
- **Povjerljivost** (engl. *Confidentiality*) – zaštita komunikacije ili pohranjenih informacija od presretanja i stavljanja na uvid neovlaštenim osobama.
- **Neporicanje** (engl. *Nonrepudiation*) – sudionici ne mogu odbiti ili poreći akciju u kojoj su sudjelovali, npr. slanje i primanje informacija.
- **Kontrola pristupa** (engl. *Access Control*) – ograničavanje pristupa informacijama i ograničavanje provođenja akcija.
- **Raspoloživost** (engl. *Availability*) – informacije moraju biti raspoložive, a sustavi i usluge u stanju operativnosti, usprkos mogućim neočekivanim i nepredvidljivim događajima, primjerice nestanku struje, prirodnim nepogodama, nesrećama i zlonamjernim napadima.

3.1. Usporedba s web aplikacijama

Iako se na prvi pogled čine sličnima, web aplikacije i web servisi se razlikuju u gotovo svakom segmentu. Web aplikacija je bilo koja aplikacija koja se nalazi na poslužitelju, a koja je namijenjena ljudima. Kako bi pružili interakciju s korisnikom, web aplikacije koriste stranice weba u svom prezentacijskom sloju. Sav korisnički unos dolazi putem prezentacijskog sloja. No, svi podaci se pohranjuju i obrađuju na poslužitelju. Web servisi se također nalaze na poslužitelju, ali primarno su namijenjeni drugim aplikacijama. Točnije, sastoje se od javnog pristupnog sučelja kojemu se pristupa standardnim HTTP metodama kao što su GET, POST i druge. Ovo ne znači da servisu nije moguće pristupiti putem web preglednika. No, većina web servisa neće imati grafičko sučelje za korisnika. Rezultati obavljanja upita se obično dobivaju u XML ili nekom drugom formatu koji je pogodan za strojnu obradu. Web servisi nude veću prilagodljivost u raznim situacijama i scenarijima. Takva prilagodljivost je potakla njihovu uporabu u raznim sustavima i ulogama. Samim time se moraju osigurati odgovarajuće mjere za njihovu zaštitu. Na primjer, većina web servisa je dostupna putem globalne mreže Internet. Kod pristupanja web servisu putem Interneta potrebno je autenticirati korisnika putem nesigurne mreže. Potrebno je osigurati mehanizme zaštite povjerljivosti, cjelovitosti i provjere pristupa prilikom pristupa servisima. No, neki web servisi su dostupni isključivo unutar intraneta neke organizacije. Tu je najčešće riječ o specifičnim servisima koji su orijentirani na neki segment poslovanja organizacije. Način autentikacije korisnika u slučaju servisa koji je dostupan putem intraneta može se razlikovati od onog u kojemu je servis dostupan svim korisnicima Interneta. Samim ograničavanjem dostupnosti usluge postigao se viši stupanj sigurnosti. U iznimnim slučajevima koriste se servisi koji su dostupni na lokalnom računalu. Kako se ovakvi sustavi mogu koristiti neovisno o mreži neće se dalje razmatrati.

Neovisno o dostupnosti web servisa, postoje određene predostrožnosti po kojima se razlikuju u odnosu na web aplikacije. Ideja računarstva zasnovanog na servisima osigurava uporabu različitih tehnologija i platformi prilikom ostvarivanja različitih funkcionalnosti. Različite platforme mogu imati drugačije sigurnosne postavke o kojima je potrebno voditi računa prilikom integracije servisa u sustav. Kod web aplikacija se obično sve radi uporabom jedne platforme. Također, potrebno je voditi računa o uparivanju sigurnosnih mehanizama. Kako su web servisi postali popularni, došlo je do velikog broja preporuka i standarda koji upotpunjuju sigurnosne aspekte servisa. Prilikom projektiranja web aplikacija, tih specifikacija i standarda je znatno manje. Radi jednostavnije implementacije sigurnosnih mehanizama, stvoreno je niz obrazaca koje je moguće primijeniti na vlastite servise. Preporučljivo je koristiti te obrasce i preporuke. Mnogi od njih se mogu primijeniti i na web aplikacije. U nastavku su opisani neki od njih, a detaljniji popis se može pronaći u dodatnoj literaturi pod [15] i [16].

- **Mehanizam obrane u dubinu** (engl. *Defense in depth*) – predstavlja dobro poznatu ideologiju koja se zasniva na uporabi većeg broja zaštitnih slojeva. Ideologija nije nastala s web servisima, a prisutna je i izvan područja informacijske sigurnosti[1]. Zasniva se na uporabi redundantnih provjera i barijera na svakom sloju posluživanja. Ukoliko jedan sloj zaštite zakaže, drugi preuzima njegov teret.
- **Rana autentikacija** (engl. *Early Authentication*) – za razliku od web aplikacija koje mogu određeni dio sadržaja nuditi korisnicima bez autorizacije, servisi često ne prate takav model. Naime, ukoliko web servis ima potrebu za autorizacijom najbolje je odmah pri prvom upitu tražiti autorizaciju korisnika.
- **Čišćenje korisničkog unosa** (engl. *Input filtering*) – ova metoda je preuzeta iz web aplikacija i ima istu svrhu za servise. Naime, sav korisnički unos se mora smatrati kao potencijalno zloćudni sadržaj. Ukoliko na sustavu postoji određena ranjivost, korisnik ju može iskoristiti uporabom posebno strukturiranih zahtjeva. Čišćenjem unosa moguće je ograničiti štetu koja se može sustavu nanijeti u slučaju ranjivosti.
- **Odrediti granice povjerenja** (engl. *Identifying trust boundaries*) – kako se sustavi zasnovani na web servisima često sastoje od više različitih podsustava, potrebno je definirati granice povjerenja. Točnije, gdje počinju i završavaju prava pristupa jednog

[1] Na primjer, prilikom projektiranja nuklearnih elektrana obavezno se koriste različiti mehanizmi obrane u dubinu. No, ideologija se može primijeniti na bilo koji sigurnosni scenarij ili proizvod.

korisničkog zahtjeva. Povjerenje može nadilaziti jedan sustav ukoliko se koriste mehanizmi poput jednostruke prijave (opisano u poglavlju 3.3.1).

- **Smanjenje mogućih točaka za izvođenje napada** (engl. *Minimizing the Attack Surface*) – ukoliko se određeno sučelje ili dio servisa više ne koristi, potrebno ga je maknuti. Naime, manjim brojem mogućih sučelja za napad smanjuje se vjerojatnost uspješnog iskorištavanja ranjivosti.
- **Nadzor korištenja** (engl. *Access Monitor*) – pratiti podatke o svakom zahtjevu koji dolazi do sustava. Takvim mehanizmom se u slučaju napada može rekonstruirati niz događaja kako bi se lakše otkrio izvor napada.

3.2. Sigurnost REST usluga

REST teorijski model je opisan u poglavlju 2.2., a u ovom poglavlju se analiziraju sigurnosne postavke REST servisa. Kako je REST model nastao kao teorijska podloga za poboljšanje učinkovitosti i jednostavniji razvoj servisa, sigurnost nije ugrađena u osnovnom modelu. Ovo predstavlja veliki problem jer prisiljava razvijatelje servisa da izmišljaju vlastite mehanizme zaštite. Sigurnosne funkcije servisa su često u drugom planu u odnosu na primarne funkcionalnosti. Ovakvi uvjeti vode povećanju ranjivih REST servisa. Prema istraživanjima organizacije CSO (dodatna literatura pod [14]), jedna od najčešćih ranjivosti REST servisa je autentifikacija korisnika uporabom jedinstvenog ključa. Iako ovaj pristup reflektira osnovno načelo REST modela, davanje jedinstvenog identifikatora svakom sredstvu, ono ima značajni nedostatak. Naime, ne koristi se tajni podatak kojime se potvrđuje identitet korisnika koji je poslao zahtjev. Drugi česti izvor ranjivosti je korištenje HTTP protokola za razmjenu korisničkih podataka za autentifikaciju. Time se lozinka prenosi u izvornom obliku kroz nepouzdanu mrežu. REST servisi često imaju iste ranjivosti kao i web aplikacije. Točnije, podložne su XSS (engl. *Cross-Site Scripting*) napadima, napadi zaobilaženja autentifikacije i CSRF (engl. *Cross-Site Request Forgery*) napadi. No, podložni su i drugim napadima koji su karakteristični samo za REST usluge. Na primjer, servisi koji koriste druge servise kako bi ostvarili određene funkcionalnosti se u nekim situacijama moraju autentificirati tom drugom servisu. Ovo se često pojavljuje u današnjim društvenim mrežama. Naime, jedna društvena mreža omogućuje stavljanje objava na korisnikov profil u drugoj društvenoj mreži. Kako bi mogla pristupit profilu korisnika koji se nalazi u domeni druge društvene mreže mora koristiti web servis te mreže. Obzirom da se za takve operacije korisnik mora autentificirati društvenoj mreži, prilikom poziva servisa potrebno je dostaviti korisnikove podatke za prijavu. Jedan od najčešćih ranjivosti je kada web servis traži od korisnika da unese svoje podatke za pristup drugoj društvenoj mreži ili servisu. Time se korisnički podaci izlažu u domeni web servisa kojemu ti podaci ne pripadaju i stvaraju niz potencijalnih prijetnji.

Iako REST modelom nisu propisane konkretne metode za zaštitu servisa moguće je upotrijebiti neke osnovne mehanizme koje se koriste u web aplikacijama. U nastavku se opisuju neke od preporuka za zaštitu REST servisa, a više informacija se može pronaći u dodatnoj literaturi pod [4] i [5].

- Ukoliko se na razini web aplikacije koristi određeno filtriranje korisničkog unosa, potrebno je iste mehanizme koristiti za web servis. Na primjer, ukoliko se izvodi filtriranje XSS unosa prilikom obrade nove poruke kroz web sučelje, potrebno je isti algoritam upotrijebiti za web servis koji omogućuje tu funkcionalnost.
- Poželjno je izbjegavati situacije u kojima se stvaraju vlastiti sigurnosni mehanizmi. Preporučuje se korištenje neke razvojne okoline ili biblioteke koja implementira provjerene mehanizme za ostvarenje sigurnosnih zahtjeva. Kako sigurnosni mehanizmi često nisu dio funkcijskih zahtjeva, u njih se ulaže znatno manje vremena što uzrokuje stvaranje ranjivih servisa. Uporabom biblioteka poput ASP.NET, Zend Framework i drugih sličnih može se na brz i učinkovit način osigurati siguran rad web servisa.
- Ukoliko se putem web servisa mogu raditi operacije pisanja, brisanja i ažuriranja stanja nije dovoljno autentificirati korisnika samo s identifikatorom. Preporučuje se korištenje lozinke kako bi se utvrdila autentičnost korisnika. U slučaju kada se koristi lozinka

potrebno je dodatno šifrirati poruku kako korisnikova lozinka ne bi bila vidljiva prilikom prijenosa kroz mrežu.

3.3. WS-* standardi za sigurnost web servisa

Uporaba uobičajenih mehanizama za zaštitu mreža često nije dovoljna kada je riječ o web servisima. Kako se radi o novom načinu interakcije s korisnicima i drugim sustavima, postoje nove prijetnje kojima je potrebno posvetiti pažnju. WS-* specifikacije su opisane u poglavlju 2.3., a u ovom poglavlju se opisuju neki standardi koji pokrivaju sigurnosne preporuke za web servise.

3.3.1. WS-Security

WS-Security specifikacija predstavlja proširenje SOAP protokola kojim se dodaju određena sigurnosna obilježja prilikom komuniciranja s web servisom[2]. Točnije, određuje kako osigurati povjerljivost i cjelovitost poruka prilikom komunikacije. Sam način ostvarenja ovih sigurnosnih zahtjeva nije strogo određen, ali su podržani razni mehanizmi. Na primjer, moguće je koristiti standardne X.509 certifikate kojima se prenose javni ključevi sudionika. Podržan je i Kerberos protokol kojime se osigurava autentikacija korisnika u nesigurnoj mreži. Za identifikaciju korisnika koriste se posebni tokeni koji jedinstveno identificiraju sudionike u komunikaciji. Kerberos podržava i povjerljivost podataka uporabom DES (engl. *Data Encryption Standard*) algoritma. Više o uporabi Kerberosa moguće je pronaći u dodatnoj literaturi pod [12]. Detalje o raspoloživim tehnikama za osiguravanje povjerljivosti i cjelovitosti putem WS-Security specifikacije moguće je pronaći u dodatnoj literaturi pod [1].

Neovisno o tehnici putem koje se ovi zahtjevi ostvaruju, WS-Security predviđa određen broj slučajeva. Ti slučajevi se opisuju u nastavku, a opisuju osnovne probleme pri osiguravanju povjerljivosti i cjelovitosti te načelne ideje kojima se rješavaju. Bitno je napomenuti kako WS-Security dodatak nije potreban ukoliko se želi osigurati samo sigurnost na razini komunikacije. Naime, sigurnosni mehanizmi transportnog sloja mrežne arhitekture, kao što je HTTPS protokol, mogu se koristiti za razmjenu SOAP poruka. HTTPS osigurava tajnost i cjelovitost poruka prilikom komunikacije. No, on nema utjecaj na poruku i njezinu interpretaciju nakon što se isporuči do konačnog cilja.

- **Identifikacija** – kada korisnik ili sustav poziva zaštićenu funkcionalnost web servisa mora predati dodatnu informaciju kojom izražava njegovo porijeklo. Ovaj postupak se često naziva potraživanje (engl. *Making a claim*). Oznaka o potraživanju se predstavlja određenom informacijom u zaglavlju SOAP poruke. WS-Security stvara dodatno zaglavlje koje oznaku o potraživanju pretvara u specifični token. Konkretan izgled tokena ovisi o tehnici koja se koristi[3].
- **Provjera autentičnosti** – potrebno je dokazati da je oznaka o potraživanju ispravna i da se radi o legitimnom korisniku.
- **Autorizacija** – nakon što se oznaka o potraživanju ovjeri potrebno je utvrditi ima li korisnik pravo zatražiti željenu uslugu.
- **Jednostruka prijava** (engl. *Single sign-on*) – jedna od složenijih sigurnosnih prepreka je kako osigurati osnovne sigurnosne zahtjeve kroz više sustava. Prilikom obavljanja određenih zadataka korisnik često mora koristiti više web servisa. Složene transakcije često koriste dodatne web servise radi bržeg odziva ili kvalitetnijeg rezultata. Kako su web servisi autonomni i neovisni jedan o drugome potreban je mehanizam kojime bi se perzistirali podatci o identitetu korisnika kroz više sustava. U suprotnom bi se korisnik morao identificirati svim sustavima ispočetka. Ideja jednostruke prijave rješava ovaj problem. Naime, uporabom metode jednostruke prijave moguće je drugim sustavima i servisima koji su uključeni u transakciju pružiti informacije koje je korisnik unio prilikom prve

[2] SOAP protokol se koristi prilikom komuniciranja s web servisima. Poglavlje 2.1. detaljno opisuje ovaj postupak.

[3] Prethodno spomenuti Kerberos protokol i X.509 certifikati su jedan od načina ostvarivanja tokena.

autorizacije. Trenutno postoje tri tehnologije koje se koriste za ostvarivanje jednostruke prijave. To su SAML (engl. *Security Assertion Markup Language*), .NET Passport i XACML (engl. *XML Access Control Markup Language*).

- **Šifriranje i digitalno potpisivanje** – ukoliko se želi zaobići uporaba HTTPS protokola za razmjenu SOAP poruka, moguće je koristiti *XML-Encryption* i *XML-Signature* dodatke. Oni predstavljaju proširenje SOAP poruka kojima se omogućuje šifriranje i digitalno potpisivanje. Time se ostvaruje povjerljivost, cjelovitost i neporecivost poruka. XML-Encryption dodatak se zasniva na šifriranju XML dokumenata i jedan je od dijelova WS-Security specifikacije. Pomoću XML-Encryption dodatka moguće je šifrirati čitavu SOAP poruku ili samo određene dijelove poput lozinke. Osiguravanje integriteta poruke moguće je uporabom XML-Signature dodatka koji ostvaruje funkcionalnosti digitalnog potpisivanja. Točnije, nudi funkcionalnosti koje omogućuju isporuku XML dokumenta s dodatnim informacijama koje predstavljaju digitalni potpis. Te informacije se oslanjaju na specifične kriptografske algoritme koje zajedno s dokumentom čine digitalni potpis. Kako je potpis vezan za dokument koji se predaje porukom, provjera potpisa osigurava da se poruka nije promijenila tijekom prijenosa.

3.3.2. WS-Trust i WS-SecureConversation

WS-Security definira osnovne mehanizme za sigurnu razmjenu poruka. WS-Trust potkategorija koristi ove osnovne mehanizme te definira dodatne opcije i dodatke za razmjenu tokena koje bi omogućilo njihovo razumijevanje kroz više različitih domena povjerenja. Naime, kako bi se osigurala komunikacije između dviju strana one moraju izmijeniti osobne podatke. No, svaka strana mora odrediti može li vjerovati podatcima od druge strane. Cilj specifikacije WS-Trust je omogućiti aplikacijama i web servisima izgradnju pouzdanih SOAP poruka. Ovo povjerenje se uspostavlja izmjenom sigurnosnih oznaka ili tokena. Dodatno, specifikacija osigurava uspostavu pouzdane veze koja je neovisna o konkretnom protokolu koji se koristi. Sigurnosne oznake je moguće izdavati, obnavljati i provjeriti neovisno o korištenom protokolu razmjene poruka.

WS-SecureConversation je proširenje WS-Security i WS-Trust specifikacija koja omogućava sigurnu komunikaciju uporabom sjedničkih ključeva. Uporabom WS-Trust mehanizama osigurava se dostava sjedničkog ključa svim stranama koje žele sigurno komunicirati. Strana koja prva inicira sigurnu komunikaciju definira koji algoritam će se koristiti te proizvodi sjednički ključ. Ovi podaci se pakiraju u SOAP poruku i šalju stranama s kojima se želi komunicirati. Naravno, moguće je dodatno šifrirati poruku. Zahvaljujući WS-Trust mehanizmima, strane koje primaju poruku s odabranim algoritmom i sjedničkim ključem znaju da se radi o povjerljivom izvoru te započinju sigurnu komunikaciju. Više informacija o WS-Trust i WS-SecureConversation moguće je pronaći u dodatnoj literaturi pod [13].

4. Sigurnosno ispitivanje web servisa

Zahvaljujući širokoj primjeni web aplikacija nastao je niz standarda i metodologija za njihovo sigurnosno ispitivanje. Većinu tih standarda i metodologija propisuje i održava organizacija OWASP. Iako OWASP nudi neke smjernice za ispitivanje sigurnosti web servisa, trenutno ne postoji niti jedna opće prihvaćena metodologija ispitivanje web servisa. No, postoje smjernice i prijedlozi kojima je moguće provjeriti određene značajke koje često uzrokuju ranjivosti kod web servisa. Jedna od takvih metodologija može se pronaći u dodatnoj literaturi pod [25]. U nastavku poglavlja se analiziraju alati i načini ispitivanja sigurnosti web servisa.

4.1. Metode sigurnosnog ispitivanja web servisa

Zbog nedostatka korisničkog sučelja, web servisi se često smatraju težom metom za obavljanje sigurnosne provjere. Web aplikacije često nude određenu strukturu kojom se međusobno povezuju njezine stranice. Ta hijerarhija se korisniku obično prikazuje kao alatna traka za navigaciju. Većina web aplikacija ima i dodatne strukture koji olakšavaju strojnu obradu hijerarhije kao što je dokument *sitemap.xml*. Ovaj dokument opisuje hijerarhijsku strukturu poveznica web aplikacije ili web stranice zapisanu u XML formatu. Enumeracija svih stranica web aplikacije je iz ovog razloga dosta jednostavna. Dovoljno je obraditi *sitemap.xml* ili ručno proći kroz alatnu traku za navigaciju. No, web servisi i sustavi koji ih pružaju nemaju ovakve dokumente. Iz tog razloga je znatno teže otkriti sva sučelja jednog sustava koji nudi web servise. Naravno, neke informacije se mogu prikupiti iz WSDL opisa sučelja, ali samo za servis kojemu pripada. Dodatno, REST servisi često nemaju ni WSDL opis što ih čini znatno težom metom. Ipak, poznate organizacije kao što su Flickr često znaju objaviti opis svog REST sučelja kako bi drugim razvijateljima olakšali pristup svojim servisima. No, ti opisi su namijenjeni ljudima i nisu pogodni za strojnu obradu. Tim se dodatno otežava proces prikupljanja informacija o web servisima koje se napada[4].

U prethodnim poglavljima su opisane tehnologije za izradu i upravljanje web servisima. Kako je većina tih tehnologija usmjerena isključivo na web servise, nastala je potreba za novim načinima ispitivanja. Naime, potrebno je ispitati sigurnosne nedostatke koji proizlaze iz uporabe tih novih tehnologija. Iz tog razloga su razvijeni alati koji omogućuju analizu SOAP poruka i drugih elemenata web servisa. Kao gradivni element web servisa, analiza SOAP poruka predstavlja ključan korak u ispitivanju sigurnosti. Detaljnom analizom SOAP poruka i WSDL opisa servisa moguće je otkriti niz informacija koje su potrebne za strukturiranje upita. Pojedini elementi SOAP poruke mogu se koristiti za pozivanje različitih funkcionalnosti. Na primjer, element naziva *getEmailAddress* može imati atribut oblika „*profileId=1000*“. Takav zahtjev bi rezultirao odgovorom u kojemu je sadržana e-mail adresa korisnika s identifikatorom 1000. Manipulacijom takvih parametara moguće je otkriti razne informacije o servisima ili korisnicima. U poglavlju 4.2.1. se detaljnije opisuje kako upotrijebiti alat Burp Suite za automatsko otkrivanje potencijalno ranjivih parametara SOAP poruke.

Iako se web aplikacije razlikuju od web servisa u velikom broj pogleda, postoje neka preklapanja kada je riječ o sigurnosti. Naime, neke ranjivosti koje su svojstvene za web aplikacije se mogu naći i u web servisima. Jedna od njih su injekcijski napadi na SOAP poruke, poznatiji kao SOAP injekcije. No, za razliku od injekcijskih napada kod web aplikacija, SOAP injekcije se znatno teže otkrivaju i iskorištavaju. Ranjivost se pojavljuje kada web servis neispravno filtrira korisnički unos koji se nalazi unutar SOAP poruke. Izvođenjem ili korištenjem tako dobivenih podataka uzrokuju se ispadi sustava koji se mogu očitovati na različite načine. Na primjer, servis u odgovoru uključuje ispis opisa greške koji napadaču može otkriti dodatne informacije o sustavu. Dodatno, može se uzrokovati izvođenje proizvoljnih naredbi na web servisu kojim je moguće manipulirati podacima na poslužitelju. Kao i kod web aplikacija, moguće posljedice iskorištavanja injekcijskih napada su nepredvidive. Upravo zato ih je važno prepoznati u što ranijoj fazi razvoja web servisa.

[4] Radi sažetosti, koraci obavljanja sigurnosne provjere (engl. *Penetration Testing*) se ne razmatraju u ovom dokumentu.

U nastavku poglavlja se opisuju neki konkretni načini pronalaženja ranjivosti web servisa, a više konkretnih primjera moguće je pronaći u dodatnoj literaturi pod [23].

4.1.1. SQL injekcije

Napad SQL injekcijom predstavlja slanje posebno strukturiranog zahtjeva koji sadrži SQL iskaz. Cilj je podmetnuti proizvoljni SQL izraz web servisu koji taj upit prosljeđuje pozadinskoj bazi podataka. Rezultat koji se time želi postići može uključivati pristup određenom skupu informacija kojima inače ne bi mogli pristupati. SQL injekcije su jedne od najsloženijih ranjivosti informacijskih sustava, a njihove posljedice mogu imati utjecaj na čitav informacijski sustav. Kako su baze podataka danas vrlo profinjene, omogućuju rane interakcije s operacijskim sustavom. Ukoliko napadač uspije iskoristiti SQL injekcijsku ranjivost za pokretanje proizvoljnih naredbi na operacijskom sustavu poslužitelja baze podataka, moguće posljedice su nepredvidive. U ranijim fazama razvoja tehnologija weba, ovaj tip ranjivosti je predstavljao velik problem. No, razvojem web aplikacija i njihovih tehnologija ova ranjivost se u velikoj mjeri na vrijeme sprječava. Nažalost, u području web servisa ovaj tip ranjivosti se javlja iznimno često. Kako se svi parametri web servisu predaju u obliku SOAP poruke, u nastavku se opisuje primjer XML datoteke koja sadrži injekcijski napad. Točnije, XML zapis se sastoji od vršnog elementa *login* koji sadrži dva podelementa *username* i *password*. Umjesto konkretno korisničkog imena unosi se SQL injekcijski niz znakova koji uzrokuje odziv u nastavku. Ovom injekcijom smo otkrili da servis koristi Microsoft SQL server bazu podataka te ODBC upravljački program za interakciju. Dodatno, dokazalo se kako web servis neispravno filtrira korisnički unos što dokazuje postojanje SQL injekcije.

```
<login>
  <username><User>SELECT * from userstable</username>
  <password>*</password>
</login>


Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft]
[ODBC SQL Server Driver][SQL Server]Syntax error Invalid string or
buffer length.
```

4.1.2. Dvostruka prijava

Web aplikacije često koriste neku vrstu sjedničkih oznaka kojima održavaju prijavu korisnika i omogućuju mu rad s povišenim ovlastima tokom te sjednice. Kada korisnik ugasi preglednik ili se odjavi, sjednica se prekida. Web servisi koriste iste metode održavanja sjednice s korisnikom. No, zbog određenih razlika u tehnologiji i arhitekturi web aplikacija i servisa, često se pojavljuju ranjivosti. Točnije, ranjivosti koje proizlaze iz neadekvatnog upravljanja korisničkom sjednicom. Ukoliko web servis neispravno održava sjednicu s korisnikom, ona je podložna napadima. Na primjer, napadi ponavljanjem ili napadi otimanjem sjednice su primjeri koji mogu nastati ukoliko ranjivost postoji. Jedan od načina na koji se ispituje ispravnost upravljanja sjednicom je metoda dvostruke prijave. Ova metoda predstavlja okviran skup koraka koji mogu dokazati postojanje ranjivosti upravljanja sjednice. U nastavku se opisuje okviran niz koraka te metode. Zbog razlika u web servisima, dokazivanje ranjivosti će u nekim slučajevima zahtijevati određenu razinu improvizacije.

- **Dvostruka prijava** – osnovna provjera dvostruke prijave je dosljedna imenu. Točnije, potrebno je prijaviti se uporabom legitimnih podataka te se ponovno prijaviti uporabom istog korisničkog računa bez da se prethodno obavila odjava. Ukoliko prijava uspije moguće je da postoji ranjivost. Dodatno, korisno je analizirati odziv na pojedini pokušaj prijave te pokušati uočiti potencijalne razlike.
- **Višestruka prijava i odjava** – u ovom koraku potrebno je obaviti višestruku prijavu i odjavu uporabom istog korisničkog računa. Točnije, obaviti prijavu zatim odjavu te taj korak ponoviti barem još jednom. Te ponovno pokušati dvostruku prijavu. Prilikom svakog odziva potrebno je analizirati razlike i sličnosti kako bi se uočili potencijalni problemi. Ukoliko se ovom provjerom ne prikupe određene neočekivane poruke o grešci, korisno je analizirati sjedničke oznake. Ponekad se sjedničke oznake proizvode na predvidiv način što može olakšati napad na sjednicu. Iako ovo nije cilj ove provjere, može ukazati na mogući napad na sjednicu korisnika.
- **Ispravan identifikator** – uporabom ispravnog korisničkog računa obavlja se prijava te dobiva sjednički ključ ili identifikator. Uporabom tako dobivenog identifikatora rade se idući zahtjevi prema web servisu. No, umjesto ispravnog identifikatora koriste se posebno strukturirani *PropertyExpansion* dodaci. SOAP poruke imaju određen skup naredbi kojima je moguće postići dinamičku obradu ili proširenje podataka. Na primjer, izraz *${=(int)(Math.random()*1000)}* će se zamijeniti nasumičnim brojem između 0 i 999. Naredba se zadaju oznakom *$* i vitičastim zagradama. Znak = označava da će izraz s desne strane rezultirati nekom vrijednošću koju treba staviti na mjesto cijele naredbe. Pozivom ugrađene naredbe *Math.random()* poziva se funkcija koja proizvodi nasumičan broj između 0 i 1. Množenjem tog rezultata s 1000, dobiva se nasumičan broj između 0 i 999. Uporabom sličnih izraza moguće je narušiti sjednicu s korisnikom ili ju zaobići. Konkretan izraz koji narušava sjednicu ovisi o načinu upravljanja sjednicama i različit je za većinu web servisa. Više informacija o PropertyExpansion izrazima moguće je pronaći u dodatnoj literaturi pod [24].
- **Zahtjevi s isteklim identifikatorima** – jedan od najčešćih napada na sjednicu je korištenje ispravnih, ali nevažećih sjedničkih oznaka ili identifikatora za obavljanje upita. U svakom koraku moguće je isprobavati razne vrijednosti za druge parametre. Na primjer, ukoliko je neki parametar povezan sa sjedničkom oznakom njegova promjena može utjecati na obradu zahtjeva.

4.1.3. Iskorištavanje poruka o grešci

Mnogi web servisi daju određenu povratnu informaciju u slučaju neispravnog unosa pojedinih parametara. Ovisno o ulozi i semantici tog parametra, moguće je iskoristiti poruke o grešci za izgradnju daljnjih napada. Na primjer, prilikom prijave korisnika moguće je u parametar koji označava korisničko ime unijeti nasumičan niz znakova. Točnije, unijeti nepostojeće korisničko ime. Ukoliko web servis javi kako takav korisnik ne postoji, takvom obradom možemo vrlo brzo enumerirati sva korisnička imena na sustavu. Time se jednostavnije dolazi do pogađanja ispravnog korisničkog računa. Odnosno, potrebno je samo pogoditi ispravnu šifru. Uobičajen način zaštite je ograničenje broja neispravnih pokušaja prijave korisnika. Ukoliko servis ne ograničava broj pokušaja, moguće je izvesti napad grubom silom (engl. *Brute-Force Attack*).

4.1.4. Upravljanje XML elementima

Iako je format SOAP poruke strogo definiran WSDL opisom, ponekad se ona ne provjerava. Točnije, u nekim slučajevima će servisi drugačije protumačiti SOAP poruku ovisno o sadržaju (ili prisutnosti) nekih elemenata. Na primjer, u nastavku je primjer poruke za prijavu korisnika kojoj je ručno dodano više *username* i *password* elemenata. Ukoliko web

servis odbije obraditi ovakvu poruku zbog neispravnog formata, ranjivosti vjerojatno nema. No, ukoliko se korisnik ipak uspješno prijavi, moguće je da postoji određeni nedostaci u tumačenju SOAP poruka. Idući korak u provjeri je izostaviti određene elemente. Na primjer, izostavljanje *password* elementa prilikom prijave. Ukoliko se korisnik uspješno autentificira servisu, očito je prisutna ranjivost zaobilaženja autentifikacije. Drugi način obavljanja ove provjere je uporaba neispravnih naziva XML elemenata. Na primjer, umjesto elementa *username* staviti proizvoljni niz znakova, kao *uname* te promatrati odziv. Dodatno, neispravno zatvaranje XML elemenata je također pogodno za provjeru ranjivosti. Cilj svih ovih metoda je utvrditi da li web servis na ispravan način tumači i prikuplja podatke iz XML elementa te mogu li se određeni koraci provjere zaobići. Web servisi često provjeravaju ispravnost ili prisutnost određenog elementa u XML zapisu poruke prije nego što ga koriste u daljnjoj logici. Ukoliko element nije prisutan izvodi se drugi niz koraka. Često se takvim grananjem može unijeti greška u logici obrade poruka i narušiti neki sigurnosnih zahtjev.

```
<login>
  <username> eviware</username>
  <username> eviware</username>
  <password> s0ApU1R0ck5</password>
  <password> s0ApU1R0ck5</password>
</login>
```

Sličan način provjere podrazumijeva unos masovnih količina podataka u jednom elementu. Na primjer, umjesto jednog korisničkog imena u elementu *username*, pokušava se unijeti niz znakova od 1000 ili više znakova. Iako se napadi preljevom spremnika ne susreću u tehnologijama za izradu servisa, ponekad je moguće uočiti neke specifičnosti kod odziva. Kako se svi zahtjevi prema web servisima zasnivaju na izmjeni XML dokumenata, moguće je umetnuti određene XPath izraze. XPath predstavlja upitni jezik koji omogućuje dohvaćanje elemenata u XML dokumentu putem posebnih izraza. Ono što jezik SQL predstavlja za baze podataka, to XPath predstavlja za XML dokumente. Primjer XPath injekcije naveden je u isječku ispod, a pospješuje isto što i SQL injekcija. Naime, osigurava da se umjesto elementa *username* i *password* ubaci SQL injekcija koja zaobilazi autentifikaciju korisnika. Radi se o dobro poznatom nizu za provjeru SQL injekcije *' OR 1=1*.

```
<login>
  string(//user[username/text()='' or '1' = '1' and
password/text()='' or '1' = '1'])
</login>
```

4.2. Alati za ispitivanje web servisa

U ovom poglavlju se analiziraju alati za ispitivanje web servisa. Neki od njih se mogu koristiti i za analizu web aplikacija, odnosno preuzeti su od njih. Iako se web servisi značajno razlikuju od web aplikacija (kako je opisano u poglavlju 3.1.), neke ranjivosti se mogu uočiti istim alatima. Uporaba alata za web aplikacije je dijelom uzrokovano manjkom specijaliziranih alata za ispitivanje sigurnosti web servisa. No, zahvaljujući popularnosti web servisa nastaje sve više i više alata za automatizirano ispitivanje.

4.2.1. Uporaba alata Burp Suite

Alat Burp Suite predstavlja jedan od najpopularnijih alata za ispitivanje sigurnosti web aplikacija. No, zbog velike prilagodljivosti može se koristiti i za ispitivanje web servisa. Kako

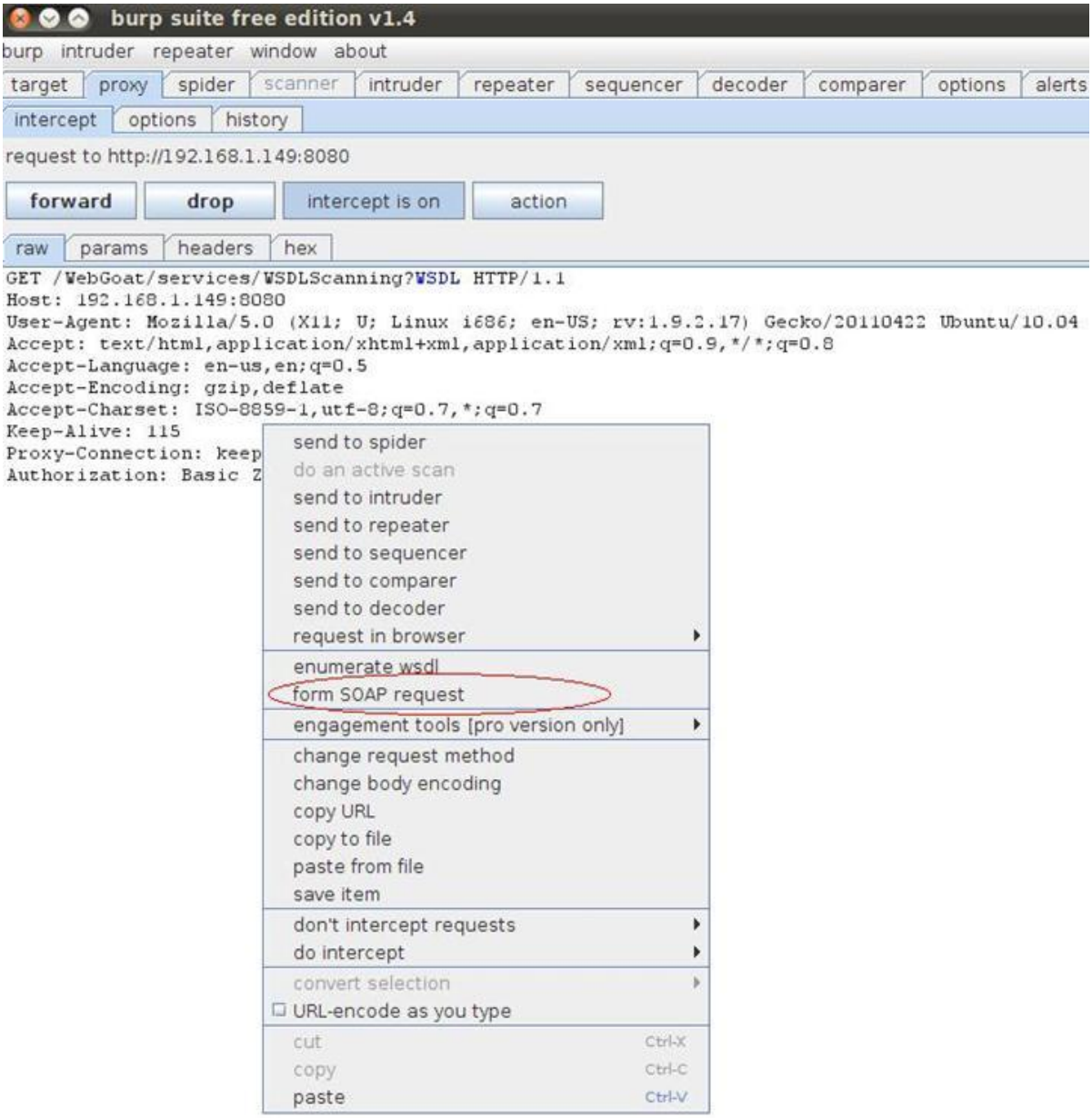
bi se izbjegla ručna analiza WSDL i SOAP poruka koje se izmjenjuju tijekom komunikacije sa servisom, moguće je koristiti Burp Suite. Iako izvorno ne podržava analizu SOAP poruka, uz manje dodatke može se prilagoditi za automatiziranu analizu web servisa. Detaljan niz koraka je opisan u dodatnoj literaturi pod [19], a u nastavku se opisuju konačni dobici uporabe ovog pristupa.

U osnovi, alatu Burp Suite se dodaje komponenta koja je sposobna interpretirati SOAP poruke. Kako je opisano u poglavlju 2.3.1, većina čvorova koji sudjeluju u komunikaciji ne znaju interpretirati SOAP poruku. Kako bi se dodala mogućnost čitanja i stvaranja SOAP poruka koriste se komponente Savon i JRuby. Programski jezik JRuby predstavlja implementaciju jezika Ruby u jeziku Java. Jezik Ruby se izvorno sastoji od niza dodataka koji nose naziv dragulj (engl. *Gem*). Savon predstavlja upravo jedan od Ruby dodataka koji može stvarati i interpretirati SOAP poruke. Koristi se JRuby inačica zbog jednostavnije integracije s alatom Burp Suite. Naime, alat Burp Suite je pisan u programskom jeziku Java. Uporabom priložene skripte (koja se nalazi u dodatnoj literaturi pod [19]) moguće je dodati funkcionalnosti automatiziranog skeniranje WSDL opisa servisa alatu Burp Suite. Kao rezultat, skripta vraća skup svih podržanih akcija web servisa te dodatne značajke o načinu rada, protokolu prijenosa i drugo. Slika 2. prikazuje rezultate jedne obrade WSDL opisa.

Osim analize WSDL opisa, moguće je na temelju njega stvoriti vlastitu SOAP poruku. Podsjetimo se, WSDL opis definira kako će izgledati SOAP poruka. Slika 3. prikazuje novo dodanu funkcionalnost stvaranja SOAP zahtjeva. Pokretanjem dodatne skripte moguće je slati proizvoljno formirane SOAP poruke web servisu te primiti i protumačiti odgovor. Budući da SOAP poruke sadrže korisnički unos koji će web servis koristiti za obračun odgovora, moguće je provjeriti postojanje injekcijskih ranjivosti uporabom standardnih metoda. Naime, injekcijski napadi se zasnivaju na unošenju posebno strukturiranih podataka na mjesto unosa. Na primjer, ukoliko se SOAP zahtjevom treba predati identifikator korisnika, pokušava se predati niz koji predstavlja moguću SQL injekciju. Ovisno o rezultatu zahtjeva, ranjivost je prisutna ili nije. Kako mogućih injekcijskih napada ima puno i za svaki od njih postoji više mogućih unosa za provjeru, predlaže se automatizirana provjera. Uporabom Intruder dodatka u Burp Suite alatu moguće je pokrenuti automatsko stvaranje zahtjeva na temelju unaprijed određenih unosa. U slučaju otkrivanja injekcijskih napada, ti unosi predstavljaju moguće parametre za provjeru postojanja ranjivosti na servisu. Korisnik može sam definirati vlastite provjere. Ipak, preporuča se korištenje unaprijed pripremljenih unosa iz baza kao što je FuzzDB. Ova biblioteka sadrži velik niz provjerenih injekcijskih ranjivosti kojima se može provjeriti postojanje ranjivosti. FuzzDB moguće je preuzeti sa [20].

Slika 2. Ispis analize WSDL opisa

Izvor: http://resources.infosecinstitute.com/soap-attack-1/



Slika 3. Stvaranje SOAP zahtjeva

Izvor: http://resources.infosecinstitute.com/soap-attack-1/

4.2.2. WSFuzzer

WSFuzzer je alat za sigurnosno ispitivanje web servisa, a održava ga organizacija OWASP (engl. *Open Web Application Security Project*). Radi se o alatu koji provjerava otpornost web servisa na neočekivani korisnički unos. Ukoliko web servis na neodgovarajući način filtrira korisnički unos, može biti podložan različitim injekcijskim i drugim vrstama napada. WSFuzzer je trenutno usmjeren na ispitivanje sigurnosti web servisa koje koriste HTTP i SOAP poruke za razmjenu podataka. Alat je napravljen s ciljem automatiziranja određenog djela provjere web servisa koji se temelji na analizi SOAP poruka. Unatoč automatiziranom ispitivanju, ovaj alat nije zamjena za ručnu analizu servisa. Naprotiv, može samo olakšati posao korisniku prilikom ispitivanja servisa. Kako je ispitivanje web servisa složen zadatak, određena razina predznanja je potrebna kako bi se mogli tumačiti rezultati analize. Razina

predznanja ovisi o konkretnim tehnologijama i standardima koji su se koristili za izradu servisa. Neki od njih su analizirani u poglavlju 3.

Iako koristan, WSFuzzer ima određena ograničenja. Jedno od njih je slaba interakcija s korisnikom prilikom izvještavanja i analize. Naime, korisnik mora sam analizirati prikupljene informacije. Alat ne nudi gotovo nikakve automatizirane metode analize prikupljenih informacija. U nastavku su nabrojane neke od korisnih funkcionalnosti ovog alata, a više informacija moguće je pronaći na službenim stranicama alata [21]. Detaljni primjeri uporabe alata za otkrivanje ranjivosti nalaze se u literaturi [22].

- Pokretanje automatiziranog napada na web servis korištenjem njegovog WSDL opisa. U slučajevima kada nedostaje WSDL opis, može se konstruirati SOAP poruka temeljem odredišne točke servisa ili drugim automatiziranim metodama. Od inačice 1.6, alat podržava i jednostavan skener za otkrivanje TCP mrežnih usluga.
- Omogućuje upravljanjem upita koji koriste višestruke parametre. Većina web servisa koristi nekoliko parametara za unos podataka. WSFuzzer olakšava provjeru svih parametara brinući se za svakog od njih. Točnije, svaki parametar se gleda kao zasebna jedinka. Od inačice 1.8.1., uvedeni su dodatni načini provjere parametara. Prvi način ispitivanja je analiza svakog parametra zasebno, jedan po jedan. Time se radi neovisna usporedba rezultata napada. Novi način rada se zasniva na istovremenom ispitivanju više parametara odjednom.
- Osnova alata je stvaranje znakovnog niza kojime bi se dokazalo postojanje ranjivosti. WSFuzzer sadrži veliku bazu znakovnih nizova koji se mogu koristiti za automatiziranu provjeru ranjivosti. Mogućnosti nadilaze provjeru običnih injekcijskih ranjivosti, ali zahtijevaju ručnu analizu rezultata. Dodatno, potrebno je savjesno koristiti ovaj alat obzirom da se uporabom određenih zloćudnih znakovnih nizova može nanijeti velika šteta sustavu koji se napada.
- Kako se automatiziranim napadima stvara velika količina mrežnog prometa postoji opasnost da se aktivira IDS (engl. *Intrusion Detection System)* sustav žrtve. WSFuzzer nudi uporabu dodatnih postavki kojima se smanjuje vjerojatnost automatskog otkrivanja napada na sustavima koji koriste IDS zaštitu.
- Ponekad se ranjivost ne može dokazati analizom rezultata, već vremenom odziva. Na primjer, prilikom slijepe provjere SQL injekcijama (engl. *Blind SQL injection*) dokaz ranjivosti se očituje razlikom u vremenu odziva. Ukoliko se odziv s injekcijskim parametrima znatno razlikuje od uobičajenog odziva, to dokazuje postojanje SQL injekcijske ranjivosti.

5. Zaključak

Web servisi su postali ključni dio u poslovanju velikog broja organizacija. Zahvaljujući brojnim pogodnostima, sve više i više organizacija počinje vlastite usluge nuditi u obliku web servisa. Nudeći vlastite web servise te korištenjem servisa drugih organizacija stvara se jedna velika interaktivna okolina. Iz tog razloga, svaki servis mora imati dobro opisano pristupno sučelje pomoću kojeg mu se pristupa. Kao jedan od prvih standarda za opis web servisa, jezik WSDL postao je ključan dio u izradi, ali i sigurnosnom ispitivanju web servisa. Kao protokol za razmjenu poruka između korisnika i servisa najčešće se koristi SOAP protokol. Izgled SOAP poruke se definira WSDL opisom, slično kao što XML shema definira izgled XML dokumenta. Iznimka je kod REST usluga koje nemaju strogo definiran protokol za razmjenu poruka s korisnikom. Dodatno, REST usluge ponekad nemaju strojno obradiv opis sučelja. Iako su web servisi relativno nova tehnologija, trenutno postoje dva različita načina njihova ostvarivanja. Točnije, arhitekture za ostvarivanje web servisa su REST i WS-Services.

No, u toj velikoj okolini web servisa moguće je naići i na one koji nude slabu razinu sigurnosti. Kako se servisi u okolini međusobno mogu koristiti i nadopunjavati, ranjivost jednog od njih može imati velik utjecaj na čitav sustav. Unatoč velikom broju preporuka i WS-* sigurnosnih specifikacija, postoji velik broj ranjivosti koje se redovito nalaze u web servisima. Mnoge od tih ranjivosti su preuzete iz web aplikacija. Dok su se te ranjivosti redovitim provjerama uspjele u većoj mjeri iskorijeniti iz svakodnevne primjene, kod web servisa su još aktivne. Jedan od razloga je i taj što trenutno ne postoji niti jedna opće prihvaćena metodologija sigurnosnog ispitivanja web servisa. Ipak, postoji velik broj smjernica i prijedloga kojima je moguće provjeriti postojanje ranjivosti kod web servisa.

Obzirom na arhitekturu, najmanju razinu sigurnosti nude REST servisi. Zbog nedostatka sigurnosnih preporuka u samom teorijskom modelu REST ideologije, ranjivosti su česta pojava. Naravno, uvijek je moguće upotrijebiti neke osnovne mehanizme zaštite koje se koriste u web aplikacijama. Unatoč ovakvom nedostatku, REST servisi su česta pojava kod mnogih organizacija. Popularne društvene mrežne kao što su Twitter, Facebook, Flickr te mnoge druge, koriste upravo REST servise.

Prilikom samog ispitivanja web servisa nedostatak grafičkog sučelja često može otežati obavljanje sigurnosne provjere. Prilikom ispitivanja web aplikacija, korisnik se susreće s velikim brojem informacija koje mogu olakšati postupak pronalaženja ranjivosti. Na primjer, alatna traka za navigaciju po web aplikaciji se može iskoristiti za enumeraciju svih funkcionalnosti i dodatnih stranica sjedišta. Ovakvih informacija prilikom ispitivanja web servisa nema budući da ne nude grafičko sučelje. Ipak, većina alata koji se koriste za sigurnosno ispitivanje web aplikacija mogu se prilagoditi za rad s web servisima. Osnovna pretpostavka je razumijevanje SOAP poruka kao osnovnog elementa za komunikaciju s web servisom. U narednim godinama se očekuje napredak u metodama ispitivanja servisa te formiranje službenih udruga koje će se baviti analizom i unaprjeđenjem novih metoda i alata za ispitivanje sigurnosti servisa, slično kao što je organizacija OWASP za web aplikacije.

6. Leksikon pojmova

CSRF (Lažiranje zahtjeva za web stranicom)

Napad na web stranice koji iskorištava ovjerenje web stranice/aplikacije prema legitimnom autoriziranom korisniku za izvođenje zlonamjernih radnji. Svrha napada je obično krađa povjerljivih informacija o autoriziranom korisniku, a napad se često dostavlja metodama društvenog inženjeringa. Točnije, žrtvi se dostavlja poveznica koja djeluje poznato, te kada korisnik otvori poveznicu pokreće se napad. - Napad' na web stranice koji iskorištava ovjerenje web stranice/aplikacije prema legitimnom autoriziranom korisniku za izvođenje zlonamjernih radnji. Svrha napada je obično krađa povjerljivih informacija o autoriziranom korisniku, a napad se često dostavlja metodama društvenog inženjeringa. Točnije, žrtvi se dostavlja poveznica koja djeluje poznato, te kada korisnik otvori poveznicu pokreće se napad.

https://www.owasp.org/index.php/Cross-Site_Request_Forgery_%28CSRF%29

DES (DES algoritam šifriranja)

Vrlo popularan kriptografski standard, danas zamjenjen standardom AES. - Vrlo popularan kriptografski standard, danas zamjenjen standardom AES. Tajni ključ za šifriranje podataka sastoji se od 56 bita, što znaći da postoji ukupno 2^56 (više od 72,000,000,000,000,000) mogućih kombinacija. Za šifriranje poruke se koristi jedan od ključeva iz velikog broja kandidata. Algoritam je simetričan, što znaći da obadvije strane moraju imati tajni ključ kako bi mogli komunicirati.

http://nvl.nist.gov/pub/nistpubs/sp958-lide/250-253.pdf

DOS napad (Napad uskraćivanjem usluge)

Napad na sigurnost na način da se određeni resurs opterećuje onemogućujući mu normalan rad. http://searchsoftwarequality.techtarget.com/definition/denial-of-service

HTTP (HyperText Transfer Protocol)

Osnovna i najčešća metoda prijenosa informacija na Webu. Predstavlja protokol na aplikacijskom sloju OSI modela, a osnovna namjena je prijenos HTML dokumenata (tj. web stranica). HTTP je request/response protokol za komunikaciju između poslužitelja i klijenta. HTTP klijent, kao što je web preglednik najčešće inicira prijenos podataka nakon što uspostavi TCP vezu s udaljenim web poslužiteljem na određenom priključku. Poslužitelj konstantno osluškuje zahtjeve na određenom mrežnom komunikacijskom priključku (tipično priljučak 80), čekajući da klijent inicira komunikaciju. - Osnovna i najčešća metoda prijenosa informacija na Webu. Predstavlja protokol na aplikacijskom sloju OSI modela, a osnovna namjena je prijenos HTML dokumenata (tj. web stranica). HTTP je request/response protokol za komunikaciju između poslužitelja i klijenta. HTTP klijent, kao što je web preglednik najčešće inicira prijenos podataka nakon što uspostavi TCP vezu s udaljenim web poslužiteljem na određenom priključku. Poslužitelj konstantno osluškuje zahtjeve na određenom mrežnom komunikacijskom priključku (tipično priključak 80), čekajući da klijent inicira komunikaciju.

http://hr.wikipedia.org/wiki/HTTP http://www.w3.org/Protocols/

Razmjerni rast (Mogućnost obrade rastuće količine zadataka)

U telekomunikacijama i programskom inženjerstvu, razmjerni rast je sposobnost sustava, mreže ili procesa da obradi rastući količinu zadataka na zadovoljavajući način, odnosno na njegovu sposobnost da bude dovoljno velik da smjesti taj porast.

http://searchdatacenter.techtarget.com/definition/scalability

SAML (Otvoreni autentikacijski standard)

Security Assertion Markup Language(SAML) je XML zasnovani otvoreni standard za razmjenu autentikacijskih i autorizacijskih podataka između sigurnosnih domena, odnosno između pružatelja identiteta i pružatelja usluga.

http://xml.coverpages.org/saml.html

SOAP (Simple Object Access Protocol)

Protokol koji služi za izmjenu strukturiranih informacija web usluga u računalnim mrežama. Za prijenos sadržaja se koristi jezik XML i koristi protokole na aplikacijskom sloju za prijenos podataka.

http://www.w3.org/TR/soap/

SQL injection napad (Napad injekcijom SQL naredbe)

Napadačka tehnika koja koristi sigurnosnu ranjivost kod pristupa web aplikacije bazi podataka. Na taj način moguće je ugroziti sigurnost web aplikacije koja konstruira SQL upite iz podataka unesenih od strane korisnika. - Napadačka tehnika koja koristi sigurnosnu ranjivost kod pristupa web programa bazi podataka. Na taj način moguće je ugroziti sigurnost web programa koji konstruira SQL upite iz podataka koje su unijeli korisnici.

https://www.owasp.org/index.php/SQL_Injection

TCP (Transmission Control Protocol)

Jedan od dva protokola usmjeravanja koja se koriste u Internetu, uspostavlja logičku vezu između krajnjih računala i osigurava pouzdani prijenos. TCP se nalazi na transportnom sloju OSI modela. - Jedan od dva protokola usmjeravanja koja se koriste u Internetu. Uspostavlja logičku vezu između krajnjih računala i osigurava pouzdani prijenos.

http://www.webopedia.com/TERM/T/TCP.html

UDDI (Universal Description, Discovery and Integration)

Standard ostvarivanja javnog imenika web servisa koji sadrži pristupna sučelja, najčešće u WSDL obliku.

http://searchsoa.techtarget.com/definition/UDDI

URI (Uniform Resource Identifier)

URI je niz znakova koji se koristi za identifikaciju imena ili nekog drugog resursa na Internetu. URI sintaksa započinje URI shemom (npr. http, ftp, mailto, sip), nakon čega slijedi dvotočka i niz znakova koji ovisi o odabranoj shemi.

http://searchsoa.techtarget.com/definition/URI

XPath (XML Path Language)

XPath predstavlja upitni jezik koji omogućuje dohvaćanje elemenata u XML dokumentu putem posebnih izraza. Ono što jezik SQL predstavlja za baze podataka, to XPath predstavlja za XML dokumente.

http://searchsoa.techtarget.com/definition/XPath

XSS napad (Cross-site scripting napad)

Napadačka tehnika koja prisiljava web aplikaciju da korisniku proslijedi zlonamjerni izvršni kod, koji se zatim učitava i izvršava u korisnikovom web pregledniku.

https://www.owasp.org/index.php/Cross-site_Scripting_%28XSS%29

WSDL (Web Services Description Language)

WSDL predstavlja dokument zasnovan na jeziku XML kojime se opisuje pristupno sučelje web servisa. U raspodijeljenim okolinama se često na temelju WSDL opisa mogu donositi određeni zaključci o korištenju (ili ne korištenu) servisa prema određenim kriterijima. WSDL opis definira izgled SOAP poruke. Točnije, koji parametri se mogu koristiti te koje metode su dostupne.

http://www.w3.org/TR/wsdl

7. Reference

[1] T. Erl, Service-Oriented Architecture (SOA): Concepts, Technology, and Design, Prentice Hall, 2005.

[2] J. Hanson, Coarse-grained interfaces enable service composition in SOA, TechRepublic, 2003.

[3] G. Alonso, F. Casati, H. Kuno, V. Machiraju, Web Services: Concepts, Architectures and Applications (Data-Centric Systems and Applications), Springer, 2010.

[4] R. Daigneau, Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services, Addison-Wesley Professional, 2011

[5] S. Allamaraju, RESTful Web Services Cookbook: Solutions for Improving Scalability and Simplicity, Yahoo Press, 2010.

[6] Web Services Addressing Working Group, 2007., http://www.w3.org/2002/ws/addr/

[7] B. Margolis, SOA for the Business Developer: Concepts, BPEL, and SCA: Concepts, BPEL and SCA, Mc Press, 2007.

[8] WS-BPEL Runtime User Guide, JBoss jBPM, http://docs.jboss.com/jbpm/bpel/v1.1/userguide/

[9] S. Tilkov, A Brief Introduction to REST, Infoq, 2007.

[10] R. Fielding, Architectural Styles and the Design of Network-based Software Architectures, doktorski rad, 2000.

[11] B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, Wiley, 1996.

[12] B. C. Neuman, T. Ts'o, Kerberos: An Authentication Service for Computer Networks, IEEE Communications Magazine, Izdanje 32, Broj 9, stranice 33-38, rujan 1994.

[13] WS-Trust 1.4, OASIS Standard, Veljača 2009., http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/ws-trust.html

[14] C. Comerford, P. Soderling, Why REST Security Doesn't Exist, CSO, 2010.

[15] M. O'Neil, Web Services Security, McGraw-Hill Osborne Media, 2003.

[16] R. Kanneganti, P. A. Chodavarapu, SOA Security, Manning Publications, 2008.

[17] T. Wilhelm, Professional Penetration Testing: Creating and Operating a Formal Hacking Lab, Syngress, 2009.

[18] J. Faircloth, C. Hurley, J. Varsalone, Penetration Tester's Open Source Toolkit, Vol. 2, Syngress, 2007.

[19] K. Johnson, Attacking Web Services Pt 1 – SOAP, InfoSecInstitute, 2011., http://resources.infosecinstitute.com/soap-attack-1/

[20] FuzzDB, 2011., http://code.google.com/p/fuzzdb/

[21] WSFuzzer, https://www.owasp.org/index.php/WSFuzzer

[22] WSFuzzer – primjer uporabe, http://www.neurofuzz.com/modules/software/vidz.php

[23] Tips and Tricks: 10 Tests of a Web Service Login you should always do, soapUI, lipanj 2010., http://www.soapui.org/SOAP-and-WSDL/web-service-hacking.html

[24] Property Expansion, soapUI, listopad 2010., http://www.soapui.org/Scripting-Properties/property-expansion.html

[25] C. Wong, D. Grzelak, A Web Services Security Testing Framework, SIFT, 2006.