Elektrotehniški vestnik 74(3): 113-118, 2007
Electrotechnical Review: Ljubljana, Slovenija

Vpliv varnostnih mehanizmov na povezljivost spletnih storitev

Simon Kocbek[1], **Matjaž B. Jurič**[2]

[1] *Fakulteta za zdravstvene vede, 2001 Maribor, Žitna ulica 15, Slovenija*
[2] *Fakulteta za elektrotehniko, računalništvo in informatiko, 2000 Maribor, Smetanova 17, Slovenija*
E-pošta: simon.kocbek@uni-mb.si, matjaz.juric@uni-mb.si

Povzetek. Pri integraciji informacijskih sistemov z uporabo spletnih storitev je treba ustrezno nasloviti področje varnosti. Prispevek proučuje povezljivost spletnih storitev v okviru standarda WSS (Web Services Security). Gre za specifikacije, ki so namenjene varovanju spletnih storitev. V prispevku smo analizirali podporo za razvoj varnih spletnih storitev na platformi Java in ogrodju .NET, proučili njihovo povezljivost ter implementirali varno storitev, izdelano v okolju Microsoft .NET in njenega odjemalca v okolju Java. Identificirali in analizirali smo probleme, ki so nastali pri njunem povezovanju.

Ključne besede: spletne storitve, integracija, WS-Security, Java, .NET

Influence of security mechanisms on web services interoperability

Extended abstract. By integration of information systems using web services, special focus has to be put on security. This article studies security mechanisms for web services. We describe use of cryptography, such as digital signature and encryption in WSS (Web Services Security) specification. We study support for secure web services development on Java platform and .NET framework, analyze interoperability, and implement a secure web service in Microsoft .NET and its client in Java. We identify and analyze the problems related to interoperability.

When we talk about security in web services, we have to think about many things. Because web services connect many nodes in a network, we have to secure every single node with limited access, identity management, password administration, authorization, etc. But special care must also be taken of securing messages that are sent between web services. They are called SOAP messages and they are constructed from XML (eXtensible Markup Language).

WSS standard dictates how to use security mechanisms in a SOAP message (figure 1). It does not say which security mechanisms to include; it just says how to include them. Because of that, the safety of a message depends on application's security requests and the proper use of WSS standard. There are several main security mechanisms that can be included in a SOAP message: security tokens, encrypted data, digital signatures and time stamps.

Security tokens are used for authentication and authorization. There are many security tokens defined and also the introduction of new ones is allowed in WSS. Three main types of security tokens are known: username tokens, binary tokens and XML security tokens.

Username security tokens are identified with <UsernameToken> tag. They are designed for including username and password information into a SOAP message header. Password can be visible or hidden. Username tokens are popular mainly because of their simplicity. Binary security tokens are identified with <BinarySecurityToken>. They can contain binary data such as digital certificates used for authentication. XML tokens do not have common identification tag as binary and username tokens do. Instead of that every XML token has its own tag.

Encryption in SOAP messages is based on XML encryption standard. It is possible to encrypt all of the content in the message or just parts of the message.

Digital signature in WSS is XML digital signature. It has two important roles. First one is to verify the validity of security tokens. For instance certificates in binary security tokens can be used for user's authentication. Because just owning a certificate and include it in a message is not enough proof of sender's identity, he must sign the certificate with his personal key. The second role of digital signature in WSS is to check and proof the integrity of the message.

With time stamps we can import time information in messages. They tell us when a message was created, when it was received and when it expires. We must take in consideration that time stamp refers to the time on the system the message was created on.

Prejet 25. avgust, 2006
Odobren 2. februar, 2007

Our thesis was that all security mechanisms mentioned above should not influence over interoperability of web services. Having that in mind, we designed an application composed of two components: a web service in .NET and its client in Java. With the application, bank account holders can use the functionality that the bank offers to them (figure 2). All the messages, sent between the service and the client, are secured with digital signature, encryption, authentication information and a time stamp.

As we discovered, WSS standard has some influence on interoperability, but all the problems were successfully solved. The main problems were default algorithms used for encryption and key generation on both platforms. Especially on Java platform, the problem was JCE (Java Cryptography Extension) package which is an algorithm framework used for encryption. The package is a part of Java SDK (Software Development Kit) and has limited support for some algorithms that are illegal in some countries (table 1). In spite of all that WSS standard itself does not interfere with web service interoperability.

Keywords: Web Services, Integration, WS-Security, Java, .NET.

1. Uvod

Prispevek obravnava področje integracije informacijskih sistemov s pomočjo spletnih storitev. Osredotoči se na varnostne mehanizme v spletnih storitvah in probleme, ki nastanejo pri njihovem povezovanju. Za razumevanje članka je zaželeno predhodno poznavanje osnov kriptografije.

Organizacije se pogosto znajdejo v položaju, ko je treba integrirati obstoječe informacijske sisteme. Informacije so danes zelo pomembne in njihova hitra izmenjava med različnimi aplikacijami je nujna. Spletne storitve so tehnologija za integracijo informacijskih virov v organizacijah in med njimi [1]. V naslednjem poglavju bomo tako pregledali, s katerimi tehnologijami so spletne storitve tesno povezane.

V tretjem poglavju bomo predstavili teoretično ozadje varnih spletnih storitev. Opisali bomo standard WSS [2]. Pomembna lastnost spletnih storitev je njihova medsebojna povezljivost. Uporaba spletne storitve mora biti neodvisna od okolja, v katerem ta teče. In varne spletne storitve niso izjema, zato bomo proučili vpliv standarda WSS na njihovo povezljivost. Tako četrto poglavje opisuje izdelavo varne spletne storitve v okolju .NET in njenega odjemalca v okolju Java. Obravnavani so problemi, na katere smo naleteli pri njunem povezovanju. Peto poglavje vsebuje sklepno misel.

2. Spletne storitve

Spletne storitve komunicirajo s pomočjo sporočil XML (eXtensible Markup Language), ki jih pošiljajo po internetu. Za njihov prenos se uporablja protokol SOAP, ki si ga lahko predstavljamo kot pisemsko ovojnico, v katero shranimo sporočilo XML pred pošiljanjem. Celotno sporočilo SOAP se nato lahko prenaša po različnih protokolih. Tako aplikacijam ni treba skrbeti, po katerem transportnem protokolu bo sporočilo poslano. Najpogosteje uporabljen protokol je protokol HTTP (Hypertext Transport Protocol). V specifikacijah SOAP do verzije 1.2 je kratica SOAP pomenila Simple Object Access Protocol. Organizacija WC3 (World Wide Web Consortium) je v tej verziji sklenila, da beseda ni več akronim, ampak le ime [3]. Vmesnik spletnih storitev je opisan z jezikom WSDL (Web Services Description Language), ki je prav tako posebna oblika jezika XML. Opisovalni jezik WSDL opiše spletno storitev kot množico operacij, ki so sestavljene iz vhodnih in izhodnih sporočil. Kot imenik vmesnikov za spletne storitve se uporablja tehnologija UDDI (Universal Description, Discovery, and Integration) [1].

Ker spletne storitve uporabljajo internet kot prenosni medij, povezujejo zelo heterogene sisteme. Aplikacije velikokrat tečejo na različnih operacijskih sistemih, razvite so z različnimi programskimi jeziki in orodji, a kljub temu se morajo sporazumeti s pomočjo nekih skupnih procedur poslovne logike in prek istega protokola. Če jim to uspe, pravimo, da so aplikacije dobro povezljive. In ravno to je lastnost spletnih storitev.

3. Varovanje spletnih storitev

Varovanje spletnih storitev je težka naloga, saj gre za različne sisteme, ki se medsebojno povezujejo. Tako ne gre le za varovanje enega sistema, ampak več vozlišč v omrežju. Na vsakem sistemu je treba poskrbeti za omejitev dostopa, upravljanje identitet, upravljanje gesel, avtorizacijo itd.

Vendar varovanje posameznih sistemov ni dovolj. Treba je poskrbeti tudi za varovanje povezav med sistemi. Ker spletne storitve komunicirajo s sporočili, je treba poskrbeti tudi za varnost le-teh. Sporočila morajo ostati nedotaknjena oziroma nespremenjena, ko potujejo med pošiljateljem in prejemnikom. Prav tako mora prejemnik ob prejetju sporočila poznati identiteto pošiljatelja. Pošiljatelju sporočila pa mora biti omogočeno, da določi pravico branja, saj mora biti neželenim osebam vpogled v vsebino sporočila onemogočen. Obstajati morajo tudi mehanizmi, ki pošiljatelju preprečijo zanikanje poslanega sporočila. Dodatno je treba poskrbeti še za varnost hranjenih sporočil. Če vsebujejo pomembne informacije (npr.: številka kreditne kartice, razne poslovne pogodbe itd.), je pomembno, da so te dobro zaščitene tudi takrat, ko se

ne pošiljajo. In točno varovanje sporočil SOAP, ne glede na to, kje se nahajajo, je naloga standarda WSS.

3.1 Standard WSS

Standard WSS je temeljni standard v okviru varovanja spletnih storitev. Temelji na specifikacijah SOAP in na standardih za digitalno podpisovanje in enkripcijo sporočil XML [5], [6]. Z upoštevanjem smernic v standardu WSS lahko sporočilo SOAP zaščitimo z naslednjimi mehanizmi:

- skrivanje informacij s prikrivanjem oziroma enkripcijo,
- doseganje verodostojnosti sporočila, preprečitev pošiljateljevega zanikanja in preverjanje celovitosti sporočila z digitalnim podpisom,
- dodajanje časovnih informacij v sporočilo s pomočjo časovnih značk.

Pred drugimi načini zaščite (npr. protokol Secure Sockets Layer oziroma SSL) ima standard WSS določene prednosti. Predvsem je pomembna njegova lastnost, da zaščitimo sama sporočila. Omenjeni protokol SSL varuje povezavo med pošiljateljem in prejemnikom, kar pomeni, da je sporočilo varno le med prenosom. Standard WSS pa zaščiti sporočilo ne glede na to, kje se nahaja in kako dolgo obstaja. Poleg tega nam specifikacije WSS omogočajo veliko fleksibilnosti: zaščitimo lahko le dele sporočil ali le vhodna sporočila, medtem ko so izhodna nezaščitena ipd. Vsako sporočilo ima svojo strategijo zaščite. Glavna pomanjkljivost standarda WSS je njegova kompleksnost, saj je odvisen od veliko drugih standardov, kot so npr. specifikacije za digitalno podpisovanje in enkripcijo sporočil XML.

Slika 1 prikazuje primer varnega sporočila SOAP. Element <Security> v zaglavju pove, da ima sporočilo vključene varnostne elemente. Element <Signature> vsebuje digitalni podpis, s pomočjo elementa <ReferenceList> se sklicujemo na šifrirane podatke, ki pa jih vsebuje element <EncryptedData>.

Ta primer predstavlja zelo poenostavljeno uporabo standarda WSS. Izpostavljeni so le najpomembnejši elementi, ki jih lahko vsebuje sporočilo. Pomembno je poudariti, da standard WSS ne narekuje katere elemente moramo vključiti v sporočilo. Pove nam le, kako elemente, ki jih želimo vključiti, vstavimo v sporočilo. Varnost posameznega sporočila je tako odvisna od zahtev aplikacije in pravilne uporabe specifikacij WSS ter standardov, ki so z povezani z njimi.

```
<Envelope>
   <Header>
      <Security>
         <!-- Varnostni žeton -->
         <UsernameToken>. . .</UsernameToken>
         <!-- XML podpis -->
         <Signature>
           <Reference URI="#Telo"/>
         </Signature>
         <!-- Reference prikrivanja v XML -->
         <ReferenceList>
            <DataReference URI="#Telo"/>
         </ReferenceList>
      </Security>
   </Header>
   <Body>
      <!-- XML prikrito telo -->
      <EncryptedData Id="Telo">
         . . .
      </EncryptedData>
   </Body>
</Envelope>
```

Slika 1. Primer varnega sporočila SOAP
Figure 1. A secure SOAP message sample

3.2 Varnostni žetoni

Varnostni žetoni so dodatni varnostni elementi v okviru sporočil SOAP, ki so namenjeni predvsem avtentikaciji ali avtorizaciji. Definiranih je veliko varnostnih žetonov, WSS pa omogoča tudi uvedbo novih.

Poznamo tri glavne tipe varnostnih žetonov: žetoni, ki vsebujejo uporabniško ime in geslo, žetoni, ki vsebujejo binarne podatke, in žetoni XML.

Žeton z uporabniškim imenom in geslom je identificiran z elementom <UsernameToken>. Omogoča preprosto vključitev uporabniškega imena in gesla v zaglavje sporočila SOAP. Geslo lahko vključimo na dva načina: prikrito ali neprikrito. Uporaba žetona z uporabniškim imenom in geslom je privlačna predvsem zato, ker je zelo preprosta za uporabnika, saj je geslo najbolj razširjena oblika avtentikacije. Če nad geslom uporabimo še zgoščevalni algoritem, je takšna oblika tudi varna. Problem je pa seveda ta, da morata strežnik in odjemalec poznati isto geslo.

Varnostni žeton, ki vsebuje binarne podatke, je označen z elementom <BinarySecurityToken>. Z njim lahko npr. v zaglavje sporočila vključimo digitalno potrdilo, potrebno za avtentikacijo.

V okviru žetonov XML pa poznamo več tipov. Vsem je skupno, da nimajo splošnega elementa, ki jih identificira (kot imajo npr. binarni žetoni element <BinarySecurityToken>), ampak ima vsak tip žetona XML svoj element [1].

3.3 Uporaba prikrivanja v sporočilu SOAP

Prikrivanje v XML nam pomaga šifrirati vsebino oziroma dele sporočil SOAP. Elementi <Envelope>, <Header> ter <Body> se ne prikrivajo, saj so potrebni za delovanje protokola SOAP. Prikriti podatki so znotraj elementov <EncryptedData>, na njih pa se sklicujemo iz zaglavja sporočila s pomočjo elementa

<ReferenceList>. Za prikrivanje lahko uporabljamo simetrične ali asimetrične algoritme, velikokrat pa se uporablja koncept digitalne ovojnice, saj se prenašajo velike količine podatkov.

3.4 Uporaba digitalnega podpisa v sporočilu SOAP

Digitalni podpis ima v WSS dve glavni nalogi [1]. Prva je preverjanje varnostnih žetonov. Kot primer opišimo podpisovanje digitalnih certifikatov. Spomnimo se, da je v WSS mogoča avtentikacija s pomočjo binarnih varnostnih žetonov v obliki certifikatov. A sam certifikat ni dovolj za avtentikacijo uporabnika, saj so ti javno dostopni in tako lahko vsak vključi poljuben certifikat v sporočilo SOAP. Zato lastnik s svojim zasebnim ključem podpiše certifikat, prejemnik sporočila pa uporabi javni ključ iz certifikata, da preveri podpis. Če je postopek uspešen, je pošiljatelj dokazal, da certifikat resnično pripada njemu, saj ima le on dostop do zasebnega ključa.

Druga naloga digitalnega podpisa v WSS je dokazovanje celovitosti samega sporočila SOAP. Če pošiljatelj podpiše telo ali katere druge dele sporočila, prejemniku omogoči, da preveri, ali je bilo sporočilo po podpisu spremenjeno.

Ker se vsebina sporočila SOAP na poti med pošiljateljem in prejemnikom velikokrat spremeni, je edini tip podpisa, ki ga je smiselno uporabiti, t.i. ločen podpis. To pomeni, da bo podpis v zaglavju sporočila, od koder se bo skliceval na podpisane podatke. Zaglavje sporočila SOAP lahko vsebuje tudi več podpisov. Tako lahko npr. dokument potuje med različnimi osebami, ki imajo možnost podpisovati različne dele dokumenta.

3.5 Časovna značka

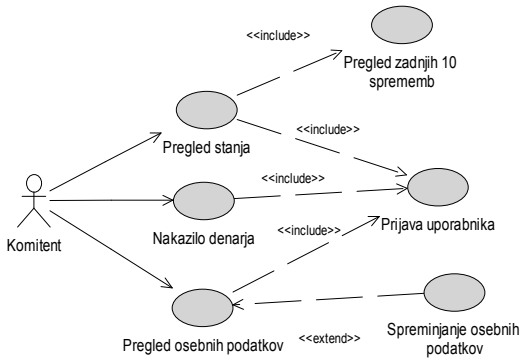
Časovna značka (angl. Time Stamp) je mehanizem, ki ga definira WSS in omogoča uvedbo časovnih informacij v sporočilo SOAP (slika 2). Tako je mogoče razbrati podatke (kdaj je sporočilo nastalo, kdaj je bilo prejeto in kdaj poteče njegova veljavnost) kar iz samega sporočila. Časovna značka se nahaja v zaglavju sporočila. Zavedati se moramo, da se čas nanaša na uro sistema, na katerem je sporočilo nastalo, in lahko pride do odstopanja med pošiljateljem in prejemnikom.

```
<Timestamp>
  <Created>
    2002-08 19T16:15:31Z
  </Created>
  <Expires>
    2002-08-19T16:20:31Z
  </Expires>
</Timestamp>
```

Slika 2: Primer časovne značke v sporočilu SOAP
Figure 2: A time stamp sample

4. Analiza povezljivosti varnih spletnih storitev

Kot smo povedali, mora biti uporaba spletne storitve neodvisna od okolja, v katerem ta teče. V ta namen smo se odločili, da bomo preučili povezljivost med dvema zelo razširjenima razvojnima okoljema: okoljem .NET in okoljem Java v okviru standarda WSS. Zamislili smo si aplikacijo, ki jo ponudi banka svojim komitentom in je sestavljena iz dveh komponent: spletne storitve ter odjemalca (slika 3).



Slika 3. Diagram primerov uporabe za aplikacijo
Figure 3. Use case diagram for the application

Komitenti lahko tako spremljajo stanje na svojem računu, spreminjajo svoje osebne podatke ter si medsebojno nakazujejo denarna sredstva. Informacije, ki se pri tem pošiljajo, so zelo občutljive, zato je scenarij primeren za demonstracijo zaščite sporočil SOAP s standardom WSS.

Posamezni koraki povezovanja odjemalca s spletno storitvijo so naslednji:

- odjemalec zaščiti sporočilo s koraki, ki so opisani niže,
- odjemalec pošlje zaščiteno sporočilo,
- spletna storitev prejme sporočilo in preveri njegovo veljavnost,
- če je sporočilo veljavno, spletna storitev pripravi odgovor,
- spletna storitev zaščiti odgovor in ga pošlje odjemalcu,
- odjemalec sprejme odgovor in preveri njegovo veljavnost.

Varnostne zahteve za vhodna sporočila spletne storitve so naslednje:

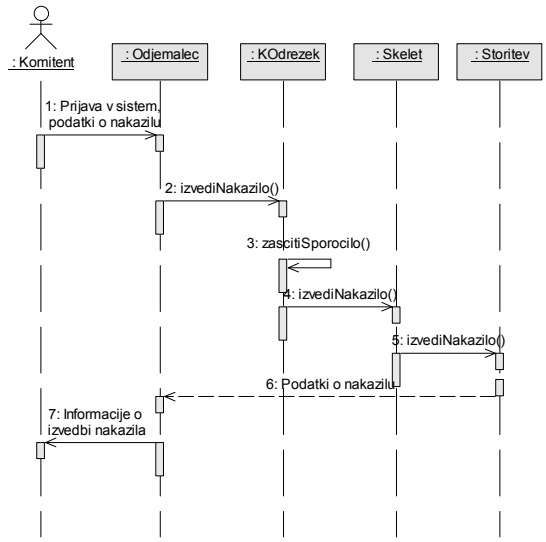
- za avtentikacijo uporabljajo varnostni žeton tipa UsernameToken z geslom ter uporabniškim imenom komitenta (geslo je v zgoščeni obliki),
- vsebovati morajo časovno značko sporočila,

- z zasebnim delom digitalnega potrdila C1 morajo biti podpisani varnostni žeton tipa UsernameToken, časovna značka in telo,
- vsebovati morajo javni del digitalnega potrdila C1 v binarnem varnostnem žetonu,
- z javnim delom digitalnega potrdila C2 mora biti prikrito telo sporočila.

Za izhodna sporočila pa mora veljati:

- vsebovati morajo časovno značko sporočila,
- z zasebnim delom digitalnega potrdila C2 morata biti podpisana časovna značka in telo,
- vsebovati morajo javni del digitalnega potrdila C2 v binarnem varnostnem žetonu,
- z javnim delom digitalnega potrdila C1 mora biti prikrito telo sporočila.

Slika 4 prikazuje diagram zaporedja za primer izvedbe nakazila. Pomembna metoda na strani odjemalca je metoda zascitiSporocilo(), ki jo kličemo nad primerkom razreda KOdrezek. Razred KOdrezek je kontrolni odrezek (Stub) v arhitekturi porazdeljenih objektov. Metoda zascitiSporocilo() poskrbi za vključitev varnostnih mehanizmov v zahtevo s pomočjo programskega vmesnika v Javi. Spletna storitev preveri veljavnost te zahteve v okviru definirane varnostne politike v okolju .NET in pošlje odgovor. Odjemalec preveri njegovo veljavnost in obvesti uporabnika o uspešnosti izvedbe nakazila.



Slika 4: Diagram zaporedja za nakazilo denarja
Figure 4: Sequence diagram for a money order

4.1 Opis tehničnih lastnosti rešitve

Za razvoj varnih spletnih storitev v okolju .NET nam podjetje Microsoft ponuja izdelek WSE (Web Services Enhancements) [7]. Zadnja verzija izdelka je 3.0. Gre za knjižnico razredov, ki omogočajo gradnjo spletnih storitev z dodatnimi možnostmi, kot je implementacija standarda WSS v ogrodj .NET s pomočjo orodja Visual Studio 2005.

Za razvoj odjemalca smo uporabili Java 5.0 SDK (Software Developer Kit) in razvojno orodje Eclipse z ustreznimi vstavki, med katerimi je najpomembnejši SD (Systinet Developer) 6.0 [8], [9]. Dodatek SD omogoča razvoj varnih spletnih storitev v okolju Java, podobno kot izdelek WSE v okolju .NET.

4.2 Testiranje in analiza povezljivosti

Po implementaciji obeh komponent rešitve smo s pošiljanjem testnih sporočil analizirali povezljivost in medsebojno delovanje odjemalca in spletne storitve. Najprej smo testirali aplikacijo brez vključene podpore standardu WSS in ugotovili, da deluje brez težav. Nato smo vključili podporo standardu WSS in pred uspešnim delovanjem aplikacije smo naleteli na nekaj problemov, ki pa smo jih uspešno odpravili. Vir vseh težav so bili različni privzeti algoritmi v obeh okoljih.

Ker se pri povezovanju spletnih storitev ponavadi pošiljajo velike količine podatkov, se pogosto uporablja koncept digitalne ovojnice [10]. To pomeni, da se za prikrivanje podatkov uporabi simetrični ključ, katerega nato prikrijemo z asimetričnim ključem. Ker so simetrični ključi kratki, se tako izognemo šifriranju dolgih sporočil. Problem je bil, da izdelek WSE 3.0 privzeto uporablja drugačen algoritem za prikrivanje simetričnega ključa kot dodatek SD 6.0. Oba sicer uporabljata algoritem RSA (Rivest Shamir Adleman), vendar različne variacije. Mogoči sta dve rešitvi problema: ali na strani spletne storitve spremenimo algoritem, ki naj ga pričakuje, ali na strani odjemalca spremenimo algoritem, ki naj ga uporabi za šifriranje. Odločili smo se za prvo možnost.

Po odpravi težave smo nadaljevali testiranje. Tokrat je spletna storitev uspešno dešifrirala sporočilo in preverila digitalni podpis, vendar se je zalomilo pri dešifriranju njenega odgovora na strani odjemalca. Razlog je bil v uporabljenem algoritmu za izračun t.i. identifikatorja ključa. Identifikator ključa je določena vrednost, s pomočjo katere identificiramo varnostni žeton in s pomočjo katere se lahko nanj tudi sklicujemo. Natančna oblika vrednosti in algoritem, ki se uporabi za njen izračun, sta odvisna od posameznega tipa varnostnega žetona. Izkazalo se je, da sta WSE in SD spet uporabljala različna privzeta algoritma, kar je na strani odjemalca povzročilo, da algoritma ni našel.

Ko smo odpravili to neskladje, je odjemalec znal dešifrirati prikriti simetrični ključ, še vedno pa z njim ni mogel dešifrirati prikritih podatkov. Napaka, ki jo je prožil, je bila v nepravilni velikosti ključa za dešifriranje. Preverili smo sporočilo in ugotovili, da je spletna storitev uporabila algoritem AES (Advanced Encryption Standard) s ključem velikosti 256 bitov za enkripcijo podatkov. Za odjemalca je bil ta ključ očitno prevelik. Kot smo ugotovili, je bil razlog v paketih JCE (Java Cryptography Extension), ki so ogrodje in

implementacija algoritmov za prikrivanje v Javi [11]. Ti paketi so nameščeni skupaj z orodjem Java SDK in ponujajo omejeno podporo algoritmom za prikrivanje. Razlog je njihov nadzor uvoza v nekaterih državah. Zato smo jih morali nadgraditi z neomejeno verzijo, da smo omogočili podporo algoritmom, ki uporabljajo ključe, večje od 128 bitov.

Tabela 1 prikazuje algoritme za prikrivanje, ki smo jih stestirali pred nagraditvijo paketa JCE.

Tabela 1: Privzeta podpora algoritmom v paketu JCE
Table 1: Default supported algorithms in JCE package

Algoritem	**Velikost ključa**	**Podpora**
AES128	128 bitov	Da
AES 256	256 bitov	Ne
AES192	192 bitov	Ne
Triple-DES (Data Encryption Standard)	168 oziroma 192 bitov	Ne

Po tej zadnji spremembi, smo končno dosegli uspešno delovanje spletne storitve in njenega odjemalca v okviru standarda WSS.

5. Sklep

V prispevku smo obravnavali varnost spletnih storitev. Te pomenijo tehnologijo, ki omogoča učinkovito integracijo sistemov v organizacijah ali med njimi.

V prvem delu prispevka smo si ogledali, kaj spletne storitve sploh so in katere tehnologije so povezane z njimi. Nato smo spoznali standard WSS, naloga katerega je definirati uporabo digitalnega podpisa in prikrivanja pri zaščiti sporočil SOAP. Seznanili smo se tudi s pojmi, kot sta varnostni žeton in časovna značka, ki dodatno pripomorejo k varnosti.

Drugi del prispevka je bil namenjen analizi povezljivosti varnih spletnih storitev. Izdelali smo spletno storitev v okolju .NET in odjemalca v okolju Java ter v obeh komponentah omogočili podporo standardu WSS. Pri njunem povezovanju smo ugotovili, da specifikacije WSS bistveno ne vplivajo na povezljivost, saj smo dosegli uspešno delovanje obeh komponent iz različnih okolij. Vendar smo pri tem naleteli na nekaj manjših težav, ki smo jih morali odpraviti.

Največji problem so bili privzeti algoritmi, ki so se na obeh sistemih razlikovali. Tako smo ugotovili, da čeprav standard WSS ponuja podporo, same platforme privzeto ne podpirajo vseh algoritmov. Zato je pomembno, da se v fazi načrtovanja varnih spletnih storitev osredotočimo tudi na proučitev teh algoritmov in njihovo podporo v razvojnih okoljih. To je pomembno tudi zato, ker se varnost algoritmov, ki se uporabljajo za šifriranje in podpisovanje, z večanjem računske moči strojne opreme manjša, kar pomeni, da moramo spletne storitve prilagajati aktualnim varnostnim mehanizmom. V nasprotnem primeru se lahko zgodi, da nekoč varna in zanesljiva spletna storitev postane nezaščitena.

6. Literatura

[1] J. Rosenberg, D. Remy, Securing Web Services with WS-Security: Demystifying WS-Security, WS-Policy, SAML, XML Signature, and XML Encryption, Sams Publishing, 2004

[2] OASIS Web Services Security Technical Committee, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss

[3] W3C, SOAP Specifications, http://www.w3.org/TR/soap/

[4] E. Newcomer, Understanding Web Services: XML, WSDL, SOAP and UDDI, Addison Wesley, 2002

[5] W3C, XML Signature Syntax and Processing, Recommendation, http://www.w3.org/TR/xmldsig-core/

[6] W3C, XML Encryption Syntax and Processing, Recommendation, http://www.w3.org/TR/xmlenc-core/

[7] Microsoft, Web Services Enhancements, http://msdn.microsoft.com/webservices/webservices/building/wse/default.aspx

[8] Eclipse, http://www.eclipse.org/

[9] Systinet, Systinet Developer, http://www.systinet.com/products/sd/overview

[10] RSA Security, What is a Digital Envelope, http://www.rsasecurity.com/rsalabs/node.asp?id=2184

[11] Sun Microsystems, Java Cryptography Extension (JCE), http://java.sun.com/products/jce/

Simon Kocbek je diplomiral leta 2006 in je mladi raziskovalec na področju računalništva in informatike na Fakulteti za zdravstvene vede Univerze v Mariboru. Izkušnje si je nabiral tudi v večjem podjetju v javni upravi. Ukvarja se z integracijo informacijskih sistemov in spletnimi aplikacijami, prav tako pa ga zanimajo področja umetne inteligence, podatkovnega rudarjenja, informatike v zdravstvu, bioinformatike in računalniške varnosti.

Matjaž B. Jurič je izredni profesor na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru. Ukvarja se s storitvenimi arhitekturami, kompozicijo poslovnih procesov, integracijo, elektronskim poslovanjem, spletnimi storitvami in optimizacijo zmogljivosti. Je avtor oz. soavtor več knjig, izdanih pri mednarodnih založbah, sodeloval je pri razvoju RMI-IIOP in je član BPEL Advisory Boarda.