1. Bezpieczeństwo transferu zestrukturalizowanych plików XML w sieci grid w oparciu o usługi Web Service poprzez protokół SOAP

Jacek Paluszak

Politechnika Gdańska,

Wydział Elektroniki, Telekomunikacji i Informatyki,

Katedra Architektury Systemów Komputerowych

e-mail: jacekpaluszak@gmail.com

Streszczenie

Niezależny protokół SOAP (ang. Simple Object Access Protocol*) działający głównie ponad protokołem HTTP (inne protokoły transportowe to np. MSMQ, MQ Series, SMTP lub TCP/IP) posiada na dzień dzisiejszy wiele rozwiązań dotyczących bezpieczeństwa transferu zestrukturalizowanych plików XML (ang.* Extensible Markup Language*). W rozdziale zaprezentowano sposoby zapobiegania nieautoryzowanym dostępom do danych przesyłanych w sieci grid przy pomocy rozproszonych komponentów udostępniających usługi Web Service poprzez protokół SOAP.*

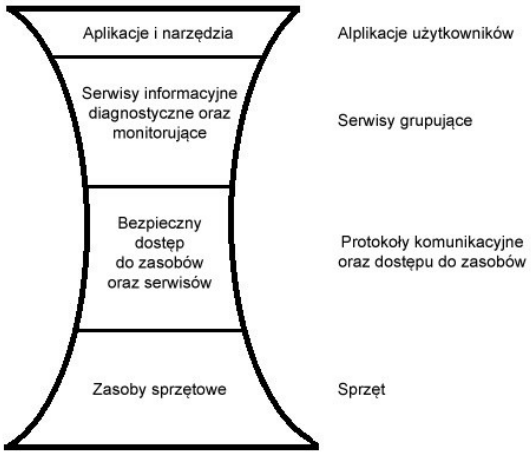
Słowa kluczowe: GRID, SOAP, Web Services, SSL/TLS, Kerberos, X.509, XML Digital Signature, XML Encryption, WS-Specifications

1.1. Wstęp

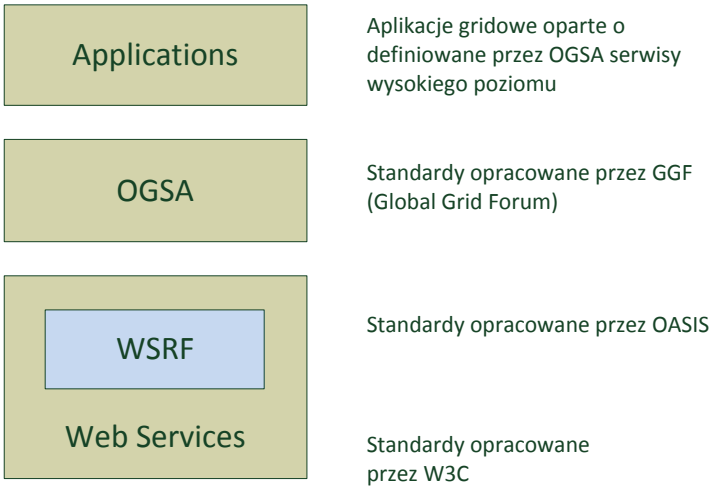
Architektura środowiska grid zdefiniowano w standardach OGSA (ang. *Open Grid Services Architecture*), OGSI *(ang. Open Grid System Infrastructure*) oraz WSRF (ang. *Web Services Resource Framework*). Standardy te określają wymagania dla interfejsu sieciowego i sposób budowania oprogramowania dla środowiska z wykorzystaniem usług sieciowych.

Grid to infrastruktura sprzętowa i programowa, która w sposób niezawodny, spójny i rozproszony zapewnia dostęp do potężnych zasobów obliczeniowych. Składa się z wielu heterogenicznych węzłów, zapewnia ściśle kontrolowane współdzielenie zasobów, integruje i steruje użytkownikami. W dużej mierze większość systemów Grid opiera się na idei Web Services. Systemy tej klasy są przezroczyste, natychmiastowo dostępne oraz osiągają wydajność porównywalną z superkomputerami przy niskim koszcie budowy.

Wysoki stopień różnorodności wykorzystywanych komponentów doprowadził do przyjęcia klepsydrowego modelu architektury systemów gridowych (patrz rys. 1.1). Paradygmat SOA (ang. *Service-Oriented Architecture*) nie pozostał bez wpływu na architekturę grid. OGSA (ang. *Open Grid Service Architecture*) stworzone przez OGF (ang. *Open Grid Forum*) to standard opisujący architekturę grid w oparciu właśnie o ten paradygmat (rys. 1.2) [40,43,58,59].



Rys. 1.1. Klepsydrowy model architektury systemów grid [40]



Rys. 1.2. Standardy architektury grid oparte na paradygmacie SOA [60]

Web Services to technologia konstrukcji rozproszonych komponentów usługowych, stanowiących podstawę dla realizacji aplikacji biznesowych w architekturze zorientowanej na usługi. Zgodnie z powszechnie akceptowaną

definicją, usługa Web Service to zwarty, samodokumentujący się komponent programowy, który może być przez swojego twórcę zarejestrowany w sieci komputerowej, a następnie przez twórcę aplikacji-konsumenta odkryty i wywołany w trybie zdalnego wykonania. Technologia Web Services opiera się na szeregu skorelowanych rozwiązań informatycznych, spośród których najważniejsze to:

- protokół komunikacyjny SOAP – służący do przekazywania zdalnych wywołań
- język opisu interfejsu usługi WSDL – służący do dystrybucji parametrów połączeń sieciowych
- specyfikacja bazy danych UDDI – służącej do rejestracji udostępnianych komponentów usługowych.

Powstało wiele specyfikacji WS-* (organizacja OASIS) składające się na architekturę GXA (ang. *Global XML Web Service Architecture*), które rozbudowują SOAP o dodatkowe funkcje zabezpieczeń dla usług Web Service :

- WS-Security
- WS-SecureConversation
- WS-Policy
- WS-SecurityPolicy
- WS-Trust
- WS-Authorization
- WS-Federation
- WS-Federation Active Requestor Profile
- WS-Federation Passive Requestor Profile [1-9,11,22,23]

1.2. Podstawowe metody uwierzytelniania

Metoda uwierzytelniania jest zdefiniowana w RFC-2617. Jest używana w tzw. *Basic* oraz *Digest Authentication* i opisana w specyfikacji HTTP. Zakłada istnienie dwóch elementów: nazwy użytkownika oraz hasła. Hasło może być przesyłane otwartym tekstem, ale może być również zakodowane w oparciu o mechanizm wykorzystywany w HTTP *Digest Authentication*, np. metodą Base64. Serwer WWW, z którym następuje połączenie, posiada listę kontroli dostępu, która pozwala również na przeprowadzenie autoryzacji. Mechanizm ten, nie jest wystarczająco bezpieczny, gdyż bardzo łatwo jest podsłuchać nazwę użytkownika i hasło.

Aby tego uniknąć stosuje się protokół SSL/TLS (zdefiniowany w RFC-2246). W protokole tym podczas nawiązywania połączenia dochodzi do wymiany certyfikatów i kluczy publicznych serwera i użytkownika oraz do uzgodnienia symetrycznego klucza, który jest następnie wykorzystywany do szyfrowania przesyłanych danych. Protokół SSL/TLS zapewnia uwierzytelnianie zarówno klienta jak i użytkownika (poprzez weryfikację certyfikatów), poufność (dzięki szyfrowaniu) oraz integralność (razem z danymi jest przesyłany również ich skrót). Nie zapewnia on jednak niezaprzeczalności, ponieważ zarówno serwer jak i użytkownik stosują ten sam klucz. W protokole SSL nie jest obowiązkowe

przesyłanie certyfikatu i klucza publicznego przez użytkownika przez co nie jest zapewnione również uwierzytelnianie klienta.

Specyfika protokołu SOAP pozwala na to, że w komunikacji pomiędzy dwiema organizacjami mogą pojawiać się pośrednicy, a u każdego z nich może zajść potrzeba zatajenia części wiadomości. Niestety, protokół SSL szyfruje cały komunikat, a dodatkowo, jako protokół warstwy transportowej, zapewnia komunikację jedynie pomiędzy dwoma węzłami, przez co umożliwia pośrednikom odczytanie całego komunikatu.

Innym sposobem uwierzytelnienia jest umieszczenie w nagłówku SOAP certyfikatu X.509, który pełni rolę żetonu zabezpieczeń identyfikującego klienta, lub osobę w imieniu której działa aplikacja kliencka. Certyfikat X.509 może być zmapowany na konkretnego użytkownika aplikacji umożliwiając tym samym określenie jego uprawnień w ramach usługi. Certyfikat X.509 jest wówczas umieszczany jako zawartość elementu *BinarySecurityToken* i kodowany w Base64.

Zgodnie ze specyfikacją WS-Security *BinarySecurityToken* może tak naprawdę przenosić w sobie 3 możliwe ładunki (ang. *payloads*):

- wsse:X509v3 – informuje, że binarnym żetonem jest certyfikat X.509 w wersji 3
- wsse:Kerberos5TGT – żetonem jest *Ticket Granting Ticket* protokołu Kerberos
- wsse:Kerberos5ST – Service Ticket protokołu Kerberos

Korzystanie z protokołu Kerberos wymaga od użytkownika podania dowodu potwierdzającego tożsamość czyli tzw. *credentials*, np. pary <nazwa użytkownika>/<hasło>, bądź certyfikatu X.509. Jeśli uwierzytelnienie przebiegnie pomyślnie klient otrzyma tzw. *Ticket Granting Ticket* (akronim TGT). Klient nie może odczytać TGT, ale może za to wykorzystać ten bilet do otrzymania tzw. *Service Ticket* (akronim ST) od *Ticket Granting Service* (akronim TGS). TGS jest usługą odpowiedzialną za przyznawanie biletów upoważniających do korzystania z konkretnych usług. Po otrzymaniu ST, klient może korzystać z danej usługi dla której przyznano mu ST.

Każda z wymienionych metod uwierzytelnienia wykorzystuje mechanizmy kryptograficzne umożliwiające podpisanie całego komunikatu, bądź wybranych fragmentów. W przypadku certyfikatu X.509 jest towarzyszący zawartemu w certyfikacie kluczowi publicznemu, klucz prywatny. W przypadku Kerberos jest to klucz sesyjny zawarty w biletach (TGT lub ST). W przypadku <nazwa użytkownika>/<hasło> jest to PBE (ang. *Based Encryption*) [9,10,12,13,14].

1.3. WS-Security, XMLDS i XMLENC

Standardy XMLDS (*XML Digital Signature*) i XMLENC (*XML Encryption*) to standardy ogólne, nie związane bezpośrednio z Web Services. Sposób ich wykorzystania do szyfrowania plików XML poprzez SOAP zdefiniowano w specyfikacji WS-Security.

Specyfikacja WS-Security nie opisuje jednego protokołu wymiany komunikatów poprzez SOAP, ale stanowi szkielet, na bazie którego można budować własne protokoły bezpieczeństwa. Umożliwia wykorzystanie dowolnych algorytmów podpisu elektronicznego i szyfrowania w celu zapewnienia uwierzytelniania, integralności, poufności i niezaprzeczalności przesyłanych komunikatów. Definiuje w nagłówku komunikatu SOAP łańcuch kolejnych transformacji i podpisów, którym podlegają fragmenty, bądź całość komunikatu, w celu zapewnienia różnych poziomów poufności, pozwalając pośrednikom na dostęp jedynie do fragmentów komunikatu przeznaczonych dla nich.

Specyfikacja XLMENC pozwala na zdefiniowanie plików XML zawierających dane zaszyfrowane dowolnym algorytmem kryptograficznym, zarówno symetrycznym (gdzie do zaszyfrowania i odszyfrowania używa się tego samego klucza) jak i asymetrycznym (gdzie używa się dwóch kluczy: publiczny do szyfrowania i prywatny do odszyfrowania wiadomości). W jednym pliku XML można również umieścić fragmenty zaszyfrowane różnymi algorytmami. Dzięki standardowi XMLENC możliwe jest konstruowanie komunikatów SOAP, w których poszczególne fragmenty mają różny poziom poufności, przykładowo część danych może zostać odczytana jedynie przez pośredników, a część jedynie przez adresata.

Specyfikacja XLMDS pozwala na zastosowanie podpisu elektronicznego do podpisywania komunikatu SOAP wywołującego usługę. Można użyć różnych algorytmów kryptografii symetrycznej jak i asymetrycznej do wygenerowania i uwierzytelnienia podpisu elektronicznego. Przy zastosowaniu kryptografii symetrycznej jest możliwość wygenerowania tzw. certyfikatu, czyli elektronicznego dokumentu, który wiąże tożsamość organizacji z jej kluczem publicznym. Dokument ten jest podpisany cyfrowo przez zaufanego administratora certyfikatów [17,19,31,32,46,27].

1.4. WS-SecureConversation

WS-SecureConversation jest to rozszerzenie WS-Security definiuje jednokrotne wzajemne uwierzytelnienie obu stron, oraz ustalenie wspólnego klucza np. symetrycznego algorytmu szyfrowania komunikacji, który będzie wykorzystywany do szyfrowania wymiany wszystkich kolejnych komunikatów SOAP aż do wygaśnięcia sesji [35].

Kryptografia klucza publicznego zapewnia bardzo wysoki poziom zabezpieczenia pod względem poufności, czy niezaprzeczalności jednak jest bardzo kosztowna pod względem przetwarzania. Dlatego w praktyce kryptografia klucza publicznego jest stosowana do szyfrowania bardzo niewielkich partii danych (do kilkudziesięciu bajtów). Pod względem wydajności szyfrowanie symetryczne przewyższa szyfrowanie asymetryczne. Z drugiej strony szyfrowanie symetryczne wymaga by po obydwu stronach znajdował się ten sam współdzielony klucz tajny.

W praktyce najczęściej stosuje się rozwiązanie hybrydowe łączące zalety obydwu rodzajów szyfrowania. Specyfikacja takiego rozwiązania została umieszczona właśnie w WS-SecureConversation.

Specyfikacja definiuje zabezpieczenie sesji składających się z więcej niż pojedynczego komunikatu. Definiuje także kontekst bezpieczeństwa (ang. *security context*) ustalany pomiędzy obydwoma stronami konwersacji. Kontekst bezpieczeństwa jest oparty o współdzielone klucze tajne (ang. *shared secrets*). Kontekst jest współdzielony przez strony konwersacji na czas trwania sesji. Na podstawie *shared secrets* generowane są sesyjne klucze tajne używane do zabezpieczania poszczególnych komunikatów.

WS-SecureConversation definiuje 3 możliwe sposoby ustalenia kontekstu bezpieczeństwa:

- ustalenie współdzielonego klucza tajnego (*shared secret*) przez zaufaną trzecią stronę – *security token service*. W takim przypadku strona inicjująca połączenie pobiera *shared secret* (żeton) i przekazuje go drugiemu uczestnikowi komunikacji.
- ustalenie *shared secret* oraz kontekstu bezpieczeństwa przez jedną ze stron komunikacji i przekazanie go stronie przeciwnej.
- negocjacja klient i usługa (znana z SSL/TLS) ustalają sposób najwłaściwszy dla obydwu stron sposób zabezpieczenia.

Czas życia kontekstu bezpieczeństwa nie zawsze pokrywa się z czasem istnienia sesji – kontekst może mieć ustalony czas ważności. Żeton kontekstu bezpieczeństwa zawiera współdzielony klucz tajny, który stanowi podstawę do generowania kluczy służących do podpisywania i szyfrowania komunikatów. Szyfrowanie każdego komunikatu odbywa się innym kluczem tajnym, który jest tworzony na podstawie klucza dotychczasowego oraz przetworzonych danych. Ten rodzaj zabezpieczenia ma zastosowanie, gdy komunikacja między klientem a usługą nie sprowadza się do jednego komunikatu – w przeciwnym razie należy się zastanowić, czy narzut związany z negocjacją klucza będzie w ogóle opłacalny wydajnościowo [17,19,31,32,46,27].

1.5. WS-Policy

Częścią metadanych opisujących interfejs (format komunikatów opisany w WSDL) mogą być również dodatkowe informacje prezentowane przez WS, które bezpośrednio nie dotyczą funkcjonalności, a wymagań WS w stosunku do klienta – jest to tzw. polityka usługi. Zadaniem tej specyfikacji jest zapewnienie szkieletu, który można wykorzystać do komunikowania przez usługę Web Services twierdzeń dotyczących jej cech i wymagań, takich jak na przykład: wymagane kodowanie przesyłanych komunikatów lub obsługiwane algorytmy podpisu elektronicznego. WS-Policy jest szkieletem, natomiast to jakie cechy usługi można komunikować definiują osobne specyfikacje zależne od domeny zastosowań.

Zgodnie z WSDL komunikaty SOAP są grupowane w operacje, które opisują podstawowe wzorce wymiany komunikatów (*request/response*). Z kolei operacje są grupowane w interfejsy – czyli zgodnie ze specyfikacją WSDL tzw. porty. Na końcu porty są wiązane z konkretnym transportem HTTP, TCP, czy SMTP. WSDL stanowi podstawę dla określenia interfejsu usługi, z której korzystają narzędzia programistyczne. Niestety opis WSDL nie jest wystarczający by ująć wszystkie możliwe cechy usługi – przykładowo informację o dostępności usługi (w określonych godzinach, czy dniach), informacje dotyczące uprawnień do korzystania z usługi (kto jest uprawniony, a kto nie jest uprawniony), zabezpieczenie komunikacji. Tego rodzaju informacje musiały być do tej pory przekazywane w sposób niezależny od samych metadanych WS i tym właśnie aspektom jest poświęcona specyfikacja WS-Policy.

Podstawą opisu każdej polityki jest tzw. asercja polityki (ang. *policy assertion*). Asercja umożliwia odpowiednie rozszerzenie metadanych w trakcie rozwoju usługi, jak również podczas, lub po wdrożeniu. Najprostszymi przykładami ograniczeń narzucanych przez asercje mogą być:

- maksymalny rozmiar komunikatu (co może mieć znaczenie np. dla urządzeń przenośnych, albo ze względów wydajnościowych)
- okresy dostępności usługi

Poszczególne asercje mogą być grupowane w alternatywy, które są wspierane przez klienta o ile spełnia on wszystkie wymagania przedstawione w alternatywie. Wsparcie alternatywy oznacza, że podmiot żądający (ang. *requestor*) spełnia wszystkie asercje wymienione w ramach tej alternatywy – co jest ustalane automatycznie poprzez ustalanie wyników poszczególnych asercji składających się na alternatywę. Cała polityka jest wspierana przez klienta o ile spełnia on warunki wymienione w przynajmniej jednej alternatywie tej polityki [16,47,48].

1.6. WS-SecurityPolicy

Jest to specyfikacja rozszerzająca WS-Policy, która definiuje twierdzenia związane z wymaganiami dotyczącymi zabezpieczenia komunikacji z daną usługą Web Service. Wprowadza ona dodatkowe asercje:

- żetony bezpieczeństwa
- integralność, poufność i czas życia komunikatów
- widoczność zawartości komunikatów dla pośredników
- ograniczenia dotyczące nagłówka bezpieczeństwa[16,47,48]

1.7. WS-Trust

Usługa może zażądać od klienta spełnienia pewnych warunków opisanych w polityce: nazwy użytkownika, klucza, uprawnień, itp. Jeśli klient nie dostarczy takich dowodów (*claims*) w komunikacie może spróbować odwołać się do odpowiedniej zaufanej trzeciej strony (*authority*), które mogłoby

dostarczyć mu odpowiednie żetony bezpieczeństwa stanowiące dowód jego uprawnień do usługi. Model bezpieczeństwa definiowany przez WS-Trust opiera się na tym, że każda usługa Web Service może wymagać, aby przychodzące żądanie zawierało podpisany żeton bezpieczeństwa.

WS-Trust jest rozszerzeniem WS-Security i obejmuje operacje dotyczące żetonów bezpieczeństwa:

- żądanie
- przyjmowanie
- wystawianie
- odnawianie
- walidację

Zaufana trzecia strona (ang. *security token service*) może być wskazana w polityce usługi. *Security token service* jest odpowiednikiem *Key Distribution Center* znanego z Kerberos. Zaufana trzecia strona może ze swojej strony również zażądać dowodów (ang. *claims*), że dany klient jest rzeczywiście uprawniony do wystawienia mu certyfikatu.

WS-Trust opisuje również protokół *challenge-response* wykorzystywany w momencie pobierania żetonu przez klienta (bądź któregoś z pośredników) – strona ubiegająca się o żeton musi w jakiś sposób udowodnić, że jest do tego uprawniona, że może być jego właścicielem. Jeśli klient, bądź jeden z pośredników dostarczy żądanych dowodów sama usługa weryfikuje, czy ufa danemu *security token service* w zakresie wystawiania żetonów danego typu. Usługa może również próbować zweryfikować dostarczone żetony u samego wystawcy (*security token service*). [33,34]

1.8. WS-Authorization, WS-Federation

Główne zadania zdefiniowane w WS-Authorization:

- uwierzytelnianie – określenie tożsamości osoby lub obiektu
- autoryzacja – określenie praw dostępu do danych, metod lub obiektów
- integralność – sprawdzenie czy dane nie zostały zmienione w drodze do celu
- podpis – sprawdzenie źródła danych
- poufność – limit użytkowników z prawami dostępu
- polityka prywatności – sprawdzanie nadużyć zasobów

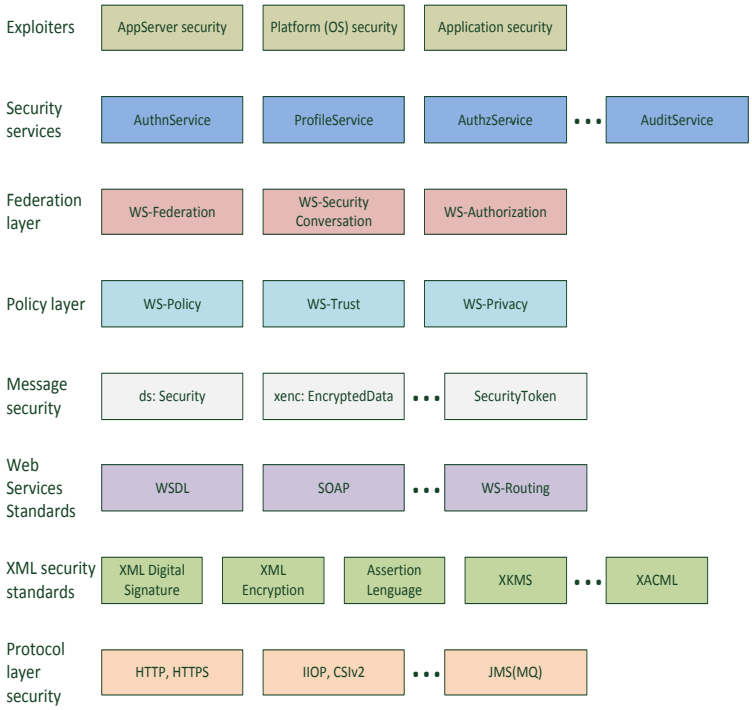
WS-Federation definiuje mechanizmy tłumaczenia odrębnych rodzajów zabezpieczeń na zgodne z wywołującym je klientem [44,61].

1.9. Podsumowanie

Bezpieczeństwo aplikacji zależy nie tylko od zastosowanych w niej standardów, ale również od projektu samej aplikacji. Nawet najlepiej implementujące standardy usługi Web Services mogą być podatne na ataki typu *SQL Injection* lub wiele innych sposobów włamań. Niebezpieczeństwo z tej

strony można zminimalizować jedynie starannie projektując usługi i przeprowadzając długie testy.

Aktualne mechanizmy (rys. 1.3) pozwalają na stworzenie dobrze zabezpieczonego środowiska grid opartego na powiązanych ze sobą usługach i aplikacjach, gdzie zaciera się różnica między aplikacjami okienkowymi a aplikacjami internetowymi. Zabezpieczenie usług tego środowiska można zintegrować niezależnie od platform i języków programowania.



Rys. 1.3. Diagram mechanizmów wykorzystywanych przy zabezpieczeniach środowiska grid poprzez Web Service [27,51,61]

Standardy bezpieczeństwa wciąż się rozwijają, tworzone są nowe specyfikacje, modyfikowane lub rozszerzane stare. Wybór odpowiedniego standardu zabezpieczeń w strukturze grid musi być uzależniony od specyfiki, charakteru i przeznaczenia systemu.

1.10. Wykaz literatury

1. http://www.ibm.com/developerworks/webservices/library/ws-add/
2. ftp://www.software.ibm.com/software/developer/library/ws-notification/WSBaseN.pdf
3. ftp://www.software.ibm.com /software/developer/library/ws-notification/WSBrokeredN.pdf
4. http://www.106.ibm.com/developerworks/library/ws-pubsub/WS-PubSub.pdf

5. http://www.106.ibm.com/developerworks/library/ws-resource/ws-modelingresources.pdf
6. http://www.106.ibm.com/developerworks/library/ws-resource/ws-resourceproperties.pdf
7. http://www.106.ibm.com/developerworks/library/wsresource/ws-resourcelifetime.pdf
8. ftp://www.software.ibm.com/software/developer/library/ws-notification/WSTopics.pdf
9. RFC 2617
10. RFC 2246
11. Nadalin A., Kaler C., Hallam-Baker P., Monzillo R. (eds), *OASIS Web Services Security: SOAP Message Security 1.0*, OASIS Standard Specification, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsssoap- message-security-1.0.pdf, 2004.
12. Public Key Infrastructure, Wikipedia, http://en.wikipedia.org/wiki/Public_key_infrastructure
13. Public Key Certificate, Wikipedia, http://en.wikipedia.org/wiki/Public_key_certificate
14. Certificate Authority, Wikipedia, http://en.wikipedia.org/wiki/Certificate_authority
15. Franks J., Hallam-Baker P., Hostetler J., Lawrence S. et al.: *RFC 2617: HTTP Authentication: Basic and Digest Access Authentication*, http://www.ietf.org/rfc/rfc2617.txt, 1999.
16. Anderson A.: *An Introduction to the Web Services Policy Language (WSPL)*, w materiałach konferencyjnych *5th IEEE International Workshop on Policies for Distributed Systems and Networks*, Yorktown Heights, New York, http://research.sun.com/projects/xacml/Policy2004.pdf, 2004
17. Larmouth J., ed.: *OASIS XML Common Biometric Format*, OASIS Standard Specification, http://www.oasis-open.org/committees/download.php/3353/oasis-200305-xcbf-specification-1.1.doc, 2003
18. Bialkowski J., Heineman K.: *OASIS Application Vulnerability Description Language v1.0*, Oasis Standard, http://www.oasis-open.org/committees/download.php/7145/AVDL%20Specification%20V1.pdf, 2004
19. Gralla P.: *THE WEB SERVICES ADVISOR: XML Firewalls*, http://searchwebservices.techtarget.com/tip/1,289483,sid26_gci855052,00.htm, 2002
20. Koch C., *The Battle for Web Services*, CIO Magazine, http://www.cio.com/archive/100103/standards.html, 2003
21. Siddiqui B.: *Web Services Security, Part 3*, http://webservices.xml.com/pub/a/ws/2003/05/13/security.html, 2003
22. Siddiqui B., *Web Services Security, Part 4*, http://webservices.xml.com/pub/a/ws/2003/07/22/security.html, 2003
23. Nadalin A., Kaler C., Monzillo R., Hallam-Baker P. (eds.): *OASIS Web Services Security: SOAP Message Security 1.1*, OASIS Standard Specification, http://www.oasisopen. org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf, 2006
24. Hallam B., Maler E. (eds.): *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)*, OASIS Standard Specification,

http://www.oasisopen. org/committees/download.php/2290/oasis-sstc-saml-1.0.zip, 2002
25. Maler E., Mishra P., Philpot R. (eds.): *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1*, OASIS Standard Specification, http://www.oasisopen. org/committees/download.php/ 3406/oasis-sstc-saml-core-1.1.pdf, 2003
26. Cantor S., Kemp J., Philpot R., Maler E. (eds.): *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0*, OASIS Standard Specification, http://docs.oasisopen. org/security/saml/v2.0/saml-core-2.0-os.pdf, 2005
27. Lockhart H.: *Demystifying Security Standards*, http://dev2dev.bea.com/ pub/a/2005/10/security_standards.html, 2005
28. Godik S., Moses T., (eds.): *OASIS eXtensible Access Control Markup Language (XACML) Version 1.0*, OASIS Standard Specification, http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf, 2003
29. Godik S., Moses T., (eds.): *OASIS eXtensible Access Control Markup Language (XACML) Version 1.1*, OASIS Committee Specification, http://www.oasis-open.org/committees/xacml/repository/cs-xacmlspecification-1.1.pdf, 2003.
30. Moses T., (ed.): *OASIS eXtensible Access Control Markup Language (XACML) Version 1.1*, OASIS Standard Specification, http://docs.oasis-open.org/ xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf, 2005
31. *XrML 2.0 Technical Overview*, http://www.xrml.org/reference/ XrMLTechnicalOverviewV1.pdf, 2002
32. Eastlake D., Reagle J., Solo D. (eds.): *XML Signature Syntax and Processing*, W3C Recommendation, http://www.w3.org/TR/xmldsig-core/, 2002
33. Eastlake D., Reagle J. (eds.): *XML Encryption Syntax and Processing*. W3C Recommendation, http://www.w3.org/TR/xmlenc-core/, 2002.
34. Rosenberg R.: *Trust, Access Control, and Rights for Web Services Part 2*, http://www.devshed.com/c/a/Security/Trust-Access-Control-and-Rights-for-Web-Services-Part-2/, 2004.
35. Anderson S., Bohren J. et al.: *Web Services Trust Language (WS-Trust)*, ftp://www6.software.ibm.com/software/developer/library/ws-trust.pdf, 2005
36. Rosenberg R.: *Trust, Access Control, and Rights for Web Services Part 1*, http://www.devshed.com/c/a/Security/Trust-Access-Control-and-Rights-for-Web-Services-Part-1/, 2004
37. *Security in a Web Services World: A Proposed Architecture and Roadmap*, ftp://www6.software.ibm.com/software/developer/library/ws-secmap.pdf, 2002
38. Anderson S., Bohren J. et al.: *Web Services Secure Conversation Language (WSSecureConversation)*, ftp://www6.software.ibm.com/software/developer/ library/ws-secureconversation.pdf, 2005.
39. *OASIS Web Services Secure Exchange (WS-SX) Technical Committee*, http://www.oasisopen. org/committees/tc_home.php?wg_abbrev=ws-sx
40. Siddiqui B.: *Web Services Security, Part 1*, http://webservices.xml.com/pub/a/ws/2003/03/04/security.html, 2003
41. Foster, C. Kesselman, (ed.): *The Grid 2: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, 2004

42. Siddiqui B., *Web Services Security, Part 2*, http://webservices.xml.com/pub/a/ws/2003/04/01/security.html, 2003
43. Bajaj S., Della-Libera G. (et al.): *Web Services Federation Language (WS-Federation)*, ftp://www6.software.ibm.com/software/developer/library/ws-fed.pdf, 2003
44. *Liberty Alliance ID-WSF 2.0 Specifications Overview*, http://xml.coverpages.org/ni2005-02-11- b.html#LA20-Specs
45. Angal R., Narayaan S., Patterson P., Simhachalam M.: *Building Identity-Enables Web Services*, http://developers.sun.com/prodtech/identserver/reference/techart/id-enabled-ws.html, 2005
46. *Liberty Alliance & WS-Federation: A Comparative Overview*, Liberty Alliance Project Wite Paper, https://www.projectliberty.org/resources/whitepapers/wsfed-liberty-overview-10-13-03.pdf, 2003
47. Drees S.: *OASIS Digital Signature Service Core Protocols, Elements, and Bindings*, 3rd OASIS Committee Draft, http://docs.oasis-open.org/dss/v1.0/dss-v1.0-spec-cd-Core-r03.pdf, 2005
48. Ford W. (et al.): *XML Key Management Specification (XKMS)*, W3C Note, http://www.w3.org/TR/xkms/, 2001
49. Bajaj S. (et al.): *Web Services Policy Framework (WS-Policy)*, http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-polfram/ws-policy-2006-03-01.pdf, 2006
50. Box D. (et al.): *Web Services Policy Assertions Language (WS-PolicyAssertions)*, ftp://www6.software.ibm.com/software/developer/library/ws-polas.pdf, 2002
51. Della-Libera G. (et al.): *Web Services Security Policy Language (WS-SecurityPolicy)*, ftp://www6.software.ibm.com/software/developer/library/ws-secpol.pdf, 2005.
52. Bajaj S. (et al.): *Web Services Policy Attachment (WS-PolicyAttachment)*, http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-polatt/ws-polat-2006-03-01.pdf, 2006
53. Moses T. (ed.): *OASIS XACML profile for Web-Services*, OASIS Working draft, http://www.oasisopen.org/committees/download.php/3661/draft-xacml-wspl-04.pdf, 2003
54. Halfhill T.: *Parallel Processing With CUDA*, Microprocessor Report, 2008
55. Challengers Consortium: *Impact of Grids*, Draft Research Agenda and Roadmap, Sixth Framework Program, IST 2005 2.5.4, 2008
56. *Technology firms in the recession, Here we go again*, The Economist, 2009
57. *TOP500 Supercomputing Sites*, http://www.top500.org/
58. Baker M.: *Cluster Computing White Paper*, University of Portsmouth, 2000
59. Snir M., Gropp W.: *MPI: The Complete Reference*, The MIT Press, 1998
60. Kesselman C., Foster I. (ed.), *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, 1998
61. Foster I.: *What is the Grid? A Three Point Checklist*, GRID today, 2002
62. Sotomayor B., Childers L.: *Globus Toolkit 4: Programming Java Services*, Morgan Kaufmann, 2005
63. Barrett B.P.: *Introduction to WS Authorization*