Proceedings of the 5th WSEAS International Conference on Telecommunications and Informatics, Istanbul, Turkey, May 27-29, 2006 (pp80-85)

Reduction of Interorganization Web Services Peers Incidents by Deployment of Asynchronous Computing Environment Profile Unification Methodology (ACEPUM)

KHALIL A. ABUOSBA, ASIM A. ELSHEIKH
Computer Information Systems Department
The Arab Academy for Banking and Financial Sciences
http://www.aabfs.org
JORDAN

Abstract: Web services are considered to be a major challenge for the information technology industry as they emerge from integration of several technologies adaptable within different architectures and platforms. Web Services are deployed within heterogeneous distributed environments; specifically B-2-B interactions are considered to be critical-mission activities for all parties involved, the main goals of these services is to provide a secured inter-organizational computing environment. We deploy web services on the web for the purpose of achieving reusability, interoperability, and standards utilizations. Web services are based on interactions of peers where loosely coupled systems interact in anonymous computing environments. WS environments are considered to be more vulnerable to faults and incidents than tightly coupled services. In this paper; we introduce a token-based methodology that is utilized for the purpose of quality assurance of the services computing environments. We introduce the Asynchronous Computing Environment Profile Unification Methodology (ACEPUM) to be used a vulnerability reduction methodology which to audit the environment profile variables. This approach utilizes the idea of reducing surface of attacks by limiting number of active resources within the environment.

Keywords: Web, services, Trust, Management, Security, agents, Vulnerability.

1. Introduction

Web Services deployment requires integration of the XML-based WSDL, SOAP, and UDDI technologies along with a dependable service container architecture. These loosely coupled technologies form the basic building block for any primitive web service; however, delivery of a web service has additional requirements. These requirements are Lifecycle Fulfillment, Characteristics Conformance, Security Attributes Fulfillment, and QoS attributes integration. Fig.1. One purpose of introducing Web Services on the web is to facilitate machine-to-machine interaction. Web Services have added an automation factor to B2B communications; it is expected that many establishments will be moving from private EDI systems to public global interoperable inexpensive Web Services as soon as reliability and high QoS are assured. As our interaction with data available on the Net demands special processing and deduction capabilities, the World Wide Web is emerging into a more semantic technologies deployment where data may be processed, shared, and reused across

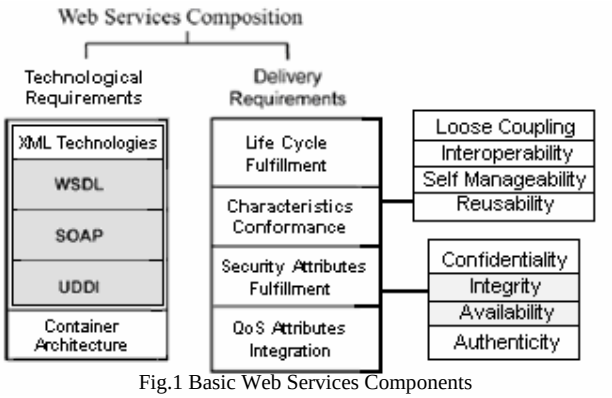

Fig.1 Basic Web Services Components

organizations and communities boundaries using metadata processing technologies. The W3C defines a Web Service as "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-

processable format (specifically WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards" [1]. A Web service is identified by a URI (Uniform Resource Identifier) whose public interfaces and bindings are defined and described using an XML component; its definition can be discovered by other software systems. Web Services are classified as Service Oriented Architectures (SOAs). SOA is based on the notion of building a software solution for an environment where service providers and requester interacts in a heterogeneous environment; this architecture is deployed for the purpose of achieving high level of abstraction, interoperability, reusability, and utilization of standardized xml-based technologies. Services built based on SOA methodology must conform to SOA conformance properties. These properties include self containment of service and self serviced, ability to invoke the service dynamically, ability to locate the service dynamically, services are network accessible, and functionalities of service and modes of operations are published as a service description document (service contract), ..etc.

1.1 Out of scope of this research

It is not the scope of this paper to build or address firewalls functionalities. We assume that all classical and traditional security breaches are installed and deployed such as firewalls, intrusion detection systems, and encryption mechanisms. We assume that peers are in conformance with the goals of secured computing environment dimensions such as information confidentiality, integrity, authenticity, and availability. It is out of scope of this research to detect or test unknown applications vulnerabilities using penetration tests or otherwise. Finally it is also, out of scope of this research address monitoring schemes of resources security.

1.2 The Scope of this research

Our focus of this research is to design an integrated vulnerability management model/methodology that aims to reduce the possibilities of occurrence of security incidents at services endpoints. We introduce the Asynchronous Computing Environment Profile Unification Methodology (ACEPUM) as a collaborative defensive measure where its functional operational goals are in conformance with SOA-based web services characteristics (listed in fig.1) deployable and interoperable in coexistence with firewalls, IDSs, and/or access control schemes resulting conveyance of trust between endpoints and increase in reliability and survivability of services.

1.3 Previous/related work

Reduction of security incidents involves vulnerability management approaches such as software vulnerabilities management and programming flaws maintenance. Sandboxing technique was introduced as a containment mechanism for executing untrusted code. It is considered to be a component based methodology that addresses seven different classes of the computing environment components ranging from device, file system, IPC, network, ptrace, signal, and including system management. This work is based on enforcement of a predefined security policy [2] that may be incorporated in our approach. Incidents management methods involve “identifying risk assessment variables” and “identifying vulnerabilities” [3]. In the field of vulnerabilities management several studies have been conducted throughout the years, a notable risk assessment methodology is the Subjective Probability Assessment [4], the technique deploy a methodology for assessing the probability of computer security incidents, some major pitfalls are involved in this technique including the possibility of faulty assumptions and predictabilities as well as environment contradictions. Alves-Foss and Barbosa introduced the System Vulnerability Index to be computed for the purpose of assessing the vulnerability of computer systems based on the system security state [5]. Some of the drawbacks of this methodology are complexity, overhead, and requirements. Incidents management paradigms where called for by Perrine and Singer such as incident management software, mobile agents for the purpose of collecting incident data and to make near real time tracing of intruders practical, and Inter-site cooperation; here, in the Inter-site cooperation, a question was raised “How can multiple sites running heterogeneous (or no) security software exchange incident data?” [6]. Mentioning these elements, the primary challenge is trust conveyance between peers or clients and servers. It is notable to say that none of the scopes or goals of the previous studies provide neither interoperability nor integration. Current deployable vulnerabilities reduction techniques involve installation of local vulnerability scanning software or remote vulnerability detection services. Both mechanisms have drawbacks and pitfalls ranging from inappropriate configuration, remote traffic exchange, high overhead, and/or time consumption, remote traffic congestion. One vantage of these solutions is that they are built based on meta-data standardized semantic languages such as OVAL, AVDL, and XCCDF.

2. Web Services Security

“WS-Security describes enhancements to SOAP messaging to provide quality of protection through message integrity, confidentiality, and authentication.

These mechanisms can be used to accommodate wide varieties of security models and encryption technologies" [7]. Several standards were defined for the purpose of securing web services transactions, these standards include WS-Security, WS-SecureConversation, WS-Trust, XML Signature, WS-Federation, Security Assertion Markup Language (SAML), Extensible Access Control Markup Language (XACML), XML Key Management Specification (XKMS), along with the standard SSL/TSL protocols. Specifically Security Attributes Fulfillment element is the core interest of our research. Confidentiality, Integrity, Availability and Authenticity are considered to be the minimal list of any secured environment, however in some cases we may drop the authenticity and confidentiality attributes due to the nature of information being processed; nevertheless availability and integrity attributes are always (in all cases) required for securing information. These two elements are compromised by two classes of flaws, the first is class is the environment vulnerabilities and the second of theses classes is the programming-bugs class. These flaws may be used by hackers and crackers to compromise the integrity and availability of information or services by several malicious attacks.

2.1 Security Incidents

Computer security incidents are defined by the US-CERT as "the acts of violating an explicit or implied security policy" [8]. In general web services environment faces the same challenges that may effect P-2-P connections and/or Client/Server architectures. Web service environment is claimed to be secured if and only if it is confidential, available and of integrity. Web services endpoints may be compromised by all standard networks threats; in addition, SOAP messages or processors are threatened by xml-based exploited vulnerabilities. Incidents are formed due to threats executions derived from either programming flaws (bugs) or environment vulnerabilities Fig.2. These threats ranges from Message interception, Man in the Middle Attacks, Session Replay Attacks, Spoofing, Denial of Services Attacks, etc.. The foremost challenge is the availability element; services must be available before worrying about their integrity or correctness; of course, this does not imply that integrity is of less importance.

2.1.1 Software Vulnerabilities & Bugs

Computing environment vulnerabilities are defined as "weaknesses in the computer system" [9]. Vulnerabilities are not considered to be harmful to systems unless they are exploited by malwares; hence, co-existence of a vulnerability and a malware is

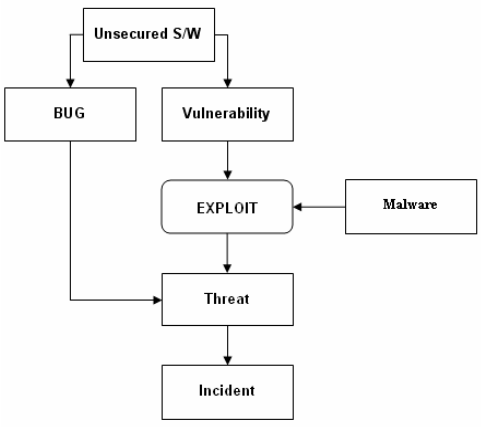

Fig.2 Incident Flow Diagram

considered to be an immediate threat to the systems that may lead to cause a violation of the security policy (incident). Another source of threats are software bugs; bugs are defined as programming flaws that may compromise a computer system. From these threes definition we conclude that incidents are results of threats executions. Threats themselves are derived from the existence of bugs and vulnerabilities. Web services computing environment compress several technologies into a single integrated communication link. These technologies may be compromised by either programming flaws (bugs) or component vulnerabilities as per fig.2. In general; security incidents involved in the web services environment are derived from attacks/flaws against one or more of the resources that composes the overall building block of a web service. Using the (CVE) list; if we abstract the web service to the semantic metadata based technologies such as XML and SOAP excluding the operating system; we will find that there are only forty nine unique XML incidents identified and only nine unique SOAP incidents listed [10]. Some of these SOAP incidents are container specific such as CVE-2004-1816 which is a candidate vulnerability that causes Denial of Service by consuming/ on client/server, peer-to-peer, group or others; they all face some common or uncommon threats depending on threat nature. Appropriate vulnerabilities management leads to threats reduction and affirms more secured computing environment. Handling Security vulnerabilities and bugs is considered to be the major obstacle in information delivery process in any computing environment within all software design methodologies. As web services are deployed based on SOA methodology and exhausting the server memory where an attacker utilizes a vulnerability associated with the container. Most of the listed SOAP vulnerabilities in the CVE list are architecture related. For example CVE-2005-2224 candidate is a platform/architecture specific. Some of the listed vulnerabilities are based on scripts

programming; again causing an existence of a threat that may lead to an incident. For any computing environment architectures whether it is based vulnerabilities management requires all the goals that SOA provides; integrating a SOA based vulnerability reduction methodology into the design of standard web services will solve many of the problems involved in securing special domain B-2-B interactions that may be standardized to all other domains including B-2-Cs.

2.1.2 Environment Vulnerabilities Rating

Web services environments vary in their specifications depending on architectural technologies and platforms being deployed. Regardless of the deployment architecture; web services computing environments are characterized as being multi-tiers end-to-end computational channels. Vulnerabilities reduction techniques also vary on each tier. Preventative measures must be taken at each endpoint. These measures may be defined in security policies. Availability of services

2.1.3 Computing Environment Operating Policy

We define this policy as an enforcement mechanism for ideal environment definition purposes. CEOP goals include verification of minimum system configuration conformance, minimization of number of active or enabled resources and services, audit latest versions installation. A major challenge arises here with a major question how to deal with platforms heterogeneity?; a first solution is to strict the interaction platforms; due to the nature of web services of being a service provider and a request interaction, we may deploy our provider peers with any platform and bind our requester peer to a certain specific platform to be specified in the WSDL contract; a second solutions would be to audit most popular environments of requester peers.

2.2 Security Maintenance Flaws

In any computing environment threats are defined as “A set of circumstances that has the potential to cause loss or harm” [9], on the other hand, bugs or programming flaws are software errors, mistakes, failures, or faults in a computer program that prevents it from working as intended, or produces an incorrect result [11]; these problems are resolved by debugging and installing up to date patches. As a quality assurance measure, an auditing of latest patches installation mechanism must be deployed prior to service initiation.

3. Vulnerabilities reduction by Asynchronous Computing Environment Profile Unification Methodology (ACEPUM)

The Peer-to-Peer (P2P) communication model is characterized by dynamic symmetry: each party exposes a set of comparable functionality and any party can initiate a communication session at any time. [12]. Securing web services computing environment involves securing endpoints as well as the communication channel. Current existing protection technologies and mechanisms such as firewalls and information encryption deployments are minimal list for securing any computing environment; however, the standard technologies lack the ability to protect the environment from already known or unknown threats. Threats may arise from unpatched software, software as well as miss-configuration is an immediate. We propose deployment of a newly hybrid message based methodology that acts as an operational trust conveyance where computing resources are audited and computing environment is minimized to its lowest enabled number of resources prior services activation. Asynchronous Computing Environment Unification Methodology calls for generating an environment identifier to be exchanged between services peers calculated based on running resources and environment identifier specified by services provider.

3.1 ACEPUM goals

Computing Environment Profile Unification Methodology aims to provide a low overhead message based solution using existing SOAP transporter. This methodology is built to be in conformance with Web Services Interoperability (WS-I) Basic Profile Version 1.0 where loose coupling is achieved. Using a predefined ideal computing environment variables defined for the purpose of minimizing the enable computing resources. Minimization of resources in operation reduces the surface of attacks on end points. Auditing these resources may be utilized as versioning checker as well as patches management tool. Auditing must be performed without using remote scanning, as remote scanning may compromise the system being scanned as well as consume some of the bandwidth of channels. The web services environment is described by using the Extensible Configuration Checklist Description Format (XCCDF) [13] specifications. Web services environment, specifically at endpoint resources are defined by using XCCDF specifications.

3.2 ACEPUM Vantages

Asynchronous Computing Environment Profile Unification Methodology aims to provide a lightweight low overhead message based solution using existing SOAP transporter. This methodology is built to be in conformance with Web Services Interoperability (WS-I) Basic Profile Version 1.0. Using a predefined ideal computing environment variables list, specified for the purpose of minimizing the number of enabled system resources. Minimization of resources leads to the

surface of attacks reduction on endpoints. Auditing these resources may be utilized as a quality assurances procedure where operational requirements are audited. Computing environments boundaries and perimeters are specified by detecting enabled computing resources and services. ACEPUM enable systems to define a more secured environment ruled by dynamic computational requirements and policies, ACEPUM may be also used as a patch management tool. Auditing must be performed without using remote scanning, as remote scanning may compromise the system being scanned. The web services environment is described by using the Extensible Configuration Checklist Description Format (XCCDF) specifications. Web services environment, specifically at endpoint resources are defined by using XCCDF specifications. ACEPUM vantages include system environment auditing at runtime prior services provision, less computational overhead, no significant consumption of network bandwidth due to the fact that scanning may be performed locally rather by a vulnerabilities scanning services provider. ACEPUM solutions may incorporate available standardized Vulnerability Assessment Language vulnerabilities management technologies such as Open (AVDL) and Application Vulnerability Description Language (AVDL) or may be integrated within these technologies. Empirically we have tested several systems with XCCDF-based auditing software (such as RU-Secure software) on a local testing environment without remote traffic involvement (local scanning) and we found these tests acceptable, however remote scanning using XCCDF-based software will be a major deployment drawback. Integrating a CEPUM model within such products will ultimately convey trust between endpoints; as the exchanged token may be used as a vehicle for authenticity, compliance and assurance.

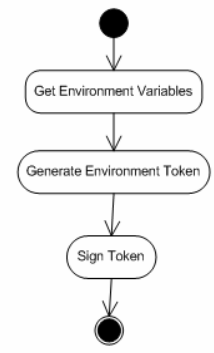

Fig.3 ACEPUM identifier generatinon process

3.3 Defining Computing Environments by Agency

In general, number of successful security incidents proportionally increases with the increment of surface of environment variables and channels; similarly by nature interoperability calls for wider area of computational operations as there are more resources to be consumed; reflecting higher rate of vulnerabilities. In any computing environment, the more we consume resources and services the more we have probability to compromise the system and, hence reducing vulnerabilities may be achieved by defining an ad-hoc computing environment. Due to the operational nature of web services we have selected a multiagent integration approach which allows us to address the auditing and benchmarking requirements which may be used as an operational trust enforcement mechanism. There are two approaches/architectures are considered to be suitable for web services environment; the first is the reactive intelligent agents architecture and the second is the belief-desire-intention agents architecture [14]. Due to goals and purposes; we incorporate the reactive agent architecture as we find it more suitable for our strict discrete deterministic environment; a looser and more flexible approach is the belief-desire-intention architecture. .

3.3.1 Environment Characteristics

Projecting properties of task environment [14] on web services computing environment; our description of an endpoint of a web service environment may be classified as an accessible, deterministic, episodic, dynamic, and discrete environment. As we have mentioned previously, our main goal of this research is to reduce software vulnerabilities by imposing limitations and rules on the computing environment; we define our operating environment to be discrete and not continuous as codes executions and management of computing resources are effected by discrete events, accessible and not inaccessible as environment variables state in web services endpoints are accessible and agents are able to obtain up to date information about the environment variables, partially observable and not fully due to remote traffic requirements for real-time/near real-time vulnerabilities management, deterministic and not stochastic due to the fact that agent functionality and environment variables state are based on predetermined definitions and declarations, episodic and not sequential for the benefit of autonomy of transaction depending on environment variables state, and finally static and not dynamic due to visibility factors as well as computational resources overhead reduction

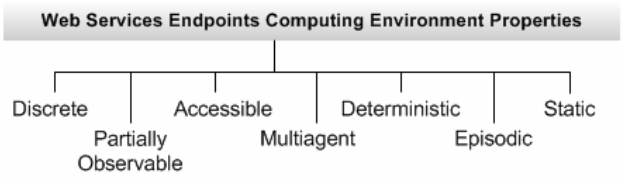

Fig.4 Proposed Ideal computing environment characteristics

3.3.2 Agents Characteristics

Our methodology proposes to incorporate a self managing two-tiered agreement negotiator as addressed by Brazier & Wijngaards [15]. A crucial element in the model is building the enforcement agent based on WS-Agreement specification. "The primary motivation for creating a service agreement between a provider and an agreement initiator is to provide assurance to the agreement initiator on the service quality and/or resource availability by the provider [16]. These agents are characterized as self managing, self modifying, pro-active, reactive, incorporates cross communications capability with other agents, autonomic, and collaborative as described in fig. 5.

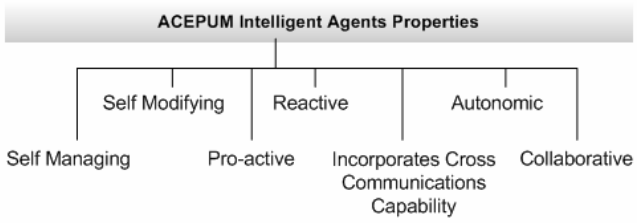

Fig. 5 Properties of ACEPUM intelligent agents

We emphasize that the agent is collaborative and autonomic due quality of service dependencies, network performance related issues as well as SOA properties conformance issues. Auditing and generating an ACEPUM identifier synchronously from a remote connection is possible, however online resources auditing involves high overhead and may cause network congestion, hence exchanging environment identifiers as small SOAP messages that may be signed and encrypted for the purposes of authenticity and confidentiality security elements.

3.3.3 Resources Auditing & Environment Benchmarking

Computing resources auditing and benchmarking are based on XCCDF specifications. A service benchmark identifier may be used as trust enforcement utility.

4. Auditing & Benchmarking

Computing resources auditing and benchmarking are based on XCCDF specifications. A service benchmark identifier may be used as trust enforcement utility.

5. Conclusion and Future Work

This methodology aims to reduce vulnerabilities in web services computing environment where it utilizes existing semantic technologies, complies by current specifications, and provides a low overhead interoperable solution without the need for remote scanning operation. In future work, we are addressing real-time/near real-time vulnerabilities updating, building an interaction model for UDDI registries for the purpose trust conveyance as well as identifying possible ontology-based solutions.

6. References

[1] W3C, W S Arch., http://www.w3.org/TR/ws-arch/

[2] Peterson, Bishop, Pandey. A Flexible Containment Mechanism for Executing Untrusted Code. In Proceedings of the 11th USENIX Security Symposium, pages 207--225, August 2002

[3] Butler, S. A. & Fischbeck, P. "Multi-Attribute Risk Assessment." SREIS 2002, 2nd Symposium on Req. Engineering for Info. Security, Raleigh, NC, October 16, 2002, CERIAS, Purdue University, Lafayette, IN.

[4] Farahmand, Navathe, Sharp, Enslow, Managing vulnerabilities of info. systems to security incidents, ACM Proceedings of the 5th international conference on E-commerce, Vol. 50, Pages: 348 – 354, 2003.

[5] Jim Alves-Foss, Salvador Barbosa: Assessing Computer Security Vulnerability. Operating Systems Review 29(3): 3-13 (1995)

[6] Perrine, Abe Singer, New paradigms in incident management, New Security Paradigms Workshop, Proceedings of the 2000 workshop on New security paradigms, Perrine and Singer,Pages: 133 - 138

[7] Web Services Security (WS-Security) Specifications,http://www-106.ibm.com/ developerworks/webservices/library/ws-secure/

[8] Common Vulnerability Exposure, US Department of Homeland Security, HTTP://www.us-cert.gov

[9] Security in Computing, 3rd Ed., Pfleeger, Prentice Hall, 2003

[10] Common Vulnerability Exposure, US Department of Homeland Security, HTTP://www.us-cert.gov

[11] WIKIPEDIA, http://en.wikipedia.org/wiki/

[12] J. Dale, D. Levine, F. McCabe, G. Arnold, M. Lyell, H. Kuno, Advanced Web Services, September 2002, White paper presented at W3C Face-to-Face in Washington, D.C.

[13] National Security Agency, Ziring, Specification for the Extensible Configuration Checklist Description Format (XCCDF), January 2005

[14] Weiss, Multiagent Systems, 3rd edition, The MIT Press, 1999, page 42-71

[15] Brazier, F.M.T., Wijngaards, Designing Self-modifying Agents in computational and cognitive Models of Creative Design V, pp. 93-112, December, 2001, University of Sydney , Gero, J.S. Maher, M.L.

[16] Dan,n Keahy, Ludwig, Rofrano, Guarantee Terms in WS-Agreement Specifications, Version 0.1, January 2004,http://www-unix.mcs.anl.gov/~keahey/ Meetings/GRAAP/WS-Agreement%20Guarantee.pdf