Australian Government
Department of Defence
Defence Science and
Technology Organisation

Service Oriented Architecture Security Risks and their Mitigation

Sarath Indrakanti

Command, Control, Communications and Intelligence Division
Defence Science and Technology Organisation

DSTO-TN-1126

ABSTRACT

Service Oriented Architecture (SOA) is an architectural paradigm and its aim is to achieve a loose coupling amongst interacting distributed systems. SOA is used by enterprises to efficiently and cost-effectively integrate heterogeneous systems. However, SOA is affected by several security vulnerabilities, thus affecting the speed of its deployment in organisations. In this report, we describe some of the security threats faced by SOA systems and corresponding risk mitigation measures by means of security technology, standards and products.

Published by

Command, Control, Communications and Intelligence Division
DSTO Defence Science and Technology Organisation
PO Box 1500
Edinburgh South Australia 5111 Australia

Telephone: (08) 7389 5555
Fax: (08) 7389 6567

AR-015-420
October 2012

APPROVED FOR PUBLIC RELEASE

Service Oriented Architecture Security Risks and their Mitigation

Executive Summary

Service Oriented Architecture (SOA) is an architectural style and its aim is to achieve a loose coupling amongst interacting components in a distributed system. SOA is designed to assist developers in building applications that interoperate and run seamlessly over heterogeneous environments deployed over multiple platforms. SOA is expected to provide benefits such as cost savings to organisations by increasing the speed of implementation of application(s) and reducing the expenditure on integration technologies [1]. However, security is one of the main roadblocks delaying deployment of SOA in organisations [2]. Computer systems and in particular distributed systems face several security risks which also affect a SOA.

In this report, we introduce some of the security threats faced by computer systems and the six security services required to counteract such threats in Section 2. The relationship between the layers of distributed systems (SOA is a type of middleware and is one of the layers) and the six security services is shown in Section 3. In Section 4, we discuss a range of security vulnerabilities faced by SOA. They include classical system vulnerabilities, Web application vulnerabilities and vulnerabilities specifically affecting SOA itself. Section 5 introduces some of the security standards created to mitigate SOA security vulnerabilities. In Section 6, we list some of the security products created to counter the threats faced by SOA. Finally, Section 7 concludes this report with some remarks.

This page is intentionally blank

Content

This page is intentionally blank

Acronyms

AVDL	Application Vulnerability Description Language
BPEL	Business Process Execution Language
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off The Shelf
CPU	Central Processing Unit
CRL	Certificate Revocation List
CSRF	Cross Site Request Forgery
DOM	Document Object Model
DOP	Defence Operations Platform
DoS	Denial of Service
DTD	Document Type Definition
ESB	Enterprise Service Bus
FIPS	Federal Information Processing Standard
HSM	Hardware Security Module
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
IPSec	Internet Protocol Security
LDAP	Lightweight Directory Access Protocol
MIME	Multipurpose Internet Mail Extensions
NRL	Naval Research Laboratory
NSA	National Security Agency
NVD	National Vulnerability Database
OASIS	Organisation for the Advancement of Structured Information Standards
OSVDB	Open Source Vulnerability Database
OWASP	Open Web Applications Security Project
P3P	Platform for Privacy Preferences
PKI	Public Key Infrastructure
SAML	Security Assertions Markup Language
SAX	Simple API for XML
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SQL	Structured Query Language
SSL	Secure Sockets Layer
SSO	Single Sign On
TCP	Transmission Control Protocol
TLS	Transport Layer Security
URL	Uniform Resource Locator
US-CERT	United States Computer Emergency Readiness Team
W3C	World Wide Web Consortium
WS	Web Service(s)
WS-BPEL	Web Services Business Process Execution Language
WSDL	Web Services Description Language
XACL	XML Access Control Language

XACML	eXtensible Access Control Markup Language
XEE	XML External Entity
XKMS	XML Key Management Specification
XML	eXtensible Markup Language
XSS	Cross Site Scripting

1. Introduction

A Service Oriented Architecture (SOA) is expected to provide benefits such as cost savings to organisations by increasing the speed of implementation of application(s) and reducing the expenditure on integration technologies [1]. However, security is one of the main roadblocks delaying deployment of SOA in organisations [2]. Computer systems and in particular distributed systems face several security risks which also affect a SOA.

In this report, we introduce some of the security threats faced by computer systems and the six security services required to counteract such threats in Section 2. The relationship between the layers of distributed systems (SOA is a type of middleware and is one of the layers) and the six security services is shown in Section 3. In Section 4, we discuss a range of security vulnerabilities faced by SOA. They include classical system vulnerabilities, Web application vulnerabilities and vulnerabilities specifically affecting SOA itself. Section 5 introduces some of the security standards created to mitigate SOA security vulnerabilities. In Section 6, we list some of the security products created to counter the threats faced by SOA. Finally, Section 7 concludes this report with some remarks.

2. Computer Systems Security

At a systems level, computer systems security is about protecting the information stored in various devices and achieving a secure transfer of information over networked devices. It pertains to the protection of data against security threats. Security threats can be divided into *passive* and *active* attacks.

Passive attacks are attacks that do not change the content of the data. They usually only monitor the data and are concerned with eavesdropping and accessing unauthorised data, and perhaps even disclosing it to others. Active attacks, on the other hand, actually change the content of the unauthorised accessed data. They make some modifications to the data or some new false data may be introduced to the existing data. Passive attacks are usually harder to detect than active attacks with the emphasis being on prevention rather than detection.

Active attacks can be further subdivided into several categories namely masquerade, replay, unauthorised access and use of data/resources, unauthorised alteration of data/resources, repudiation of actions and unauthorised denial of service. Masquerade, as the name suggests, is the pretence of one entity to be another to gain access to some data or resources. Replay is the passive capture of data and its subsequent reuse to produce an unauthorised effect. Unauthorised access, as the name suggests, is an illegal action that leads to the use of any resources such as files, application programs, computer memory, operating systems and databases. The unauthorised alteration of data implies some or all parts of legitimate and correct data are illegitimately modified or removed illegally, or the insertion or fabrication of completely new incorrect data. The repudiation of actions occurs when a party denies that it has performed that action, even though it has; it is the threat pertaining to accountability. An

unauthorised denial of service occurs when one party denies access to resources (such as a website) to entities that are otherwise authorised to use them.

A range of security services is available for computer systems, which can be leveraged to counteract the threats imposed by these types of attacks.

An **Authorisation** or access control service limits and controls access to data or resources. The authorisation service controls access by the subjects to the resources based on access control policies. Access control policies can be expressed as access control lists, capability tickets or security labels.

An **Authentication** service identifies an entity attempting to access a system. This service ensures that the originator of a request is the rightful originator and is not an impostor masquerading as the originator. An authentication service may employ a simple mechanism such as a user identity and password scheme or more sophisticated key based mechanisms such as such as Kerberos [3] (based on symmetric keys) or X.509 [4] (based on public keys) to authenticate the users.

A **Confidentiality** service protects against the unauthorised disclosure of data or resources. The service employs cryptographic mechanisms to protect against unauthorised disclosure of information.

An **Integrity** service protects against the unauthorised alteration of data or resources. The service employs cryptographic protocols and chaining techniques such as message authentication codes and hash functions to counteract such threats.

A **Non-repudiation** service protects against the repudiation of actions. The service relies on a trusted third party for the arbitration of disputes. It provides the proof that certain action has taken place. It can protect the originator of data against false denial by the recipient as well as a recipient against the false denial by the originator. A non-repudiation service must be in place prior to the information transfer. Digital signatures are used as a common mechanism to achieve non-repudiation.

An **Auditing** service, although not directly involved in the prevention of security violations assists in their detection. It tests the adequacy of the other security services in place and the conformance of the system to the security policy. Common mechanisms include the definition of the security-related events to be audited, the definition of the audit record, and the definition and generation of security alarms and actions. Also, audit trails can be stored and analysed at regular intervals. It is important to note that auditing services themselves require confidentiality, integrity and authentication services for their protection.

Security Policy Management: All the security services described above have related security information such as keys, rules or policies. Depending on the type of security service, security authorities can be defined which are responsible for managing such security information. It is important to note that the security management functions themselves should be secured from the types of threats discussed above.

3. Distributed Systems Security

Figure 1 shows the marriage between security services and distributed systems and is referred to as the UT model in [5], because it resembles the shape of the alphabet letters U and T. The legs of the U are an abstract representation of a distributed system. The six security services and their management are represented by the T. Distributed systems assume an underlying fully functional network. Every node in a distributed system runs essentially on a piece of hardware. On top of the hardware sits an operating system, which in turn is a platform for deploying middleware and for running multiple applications. Finally, users make use of various types of applications to perform a number of tasks. In general, all the six security services may be required across the layers of a distributed system. The UT model is used to highlight the various design choices and trade offs in determining whether, where and how security services should be designed and integrated into a distributed system.

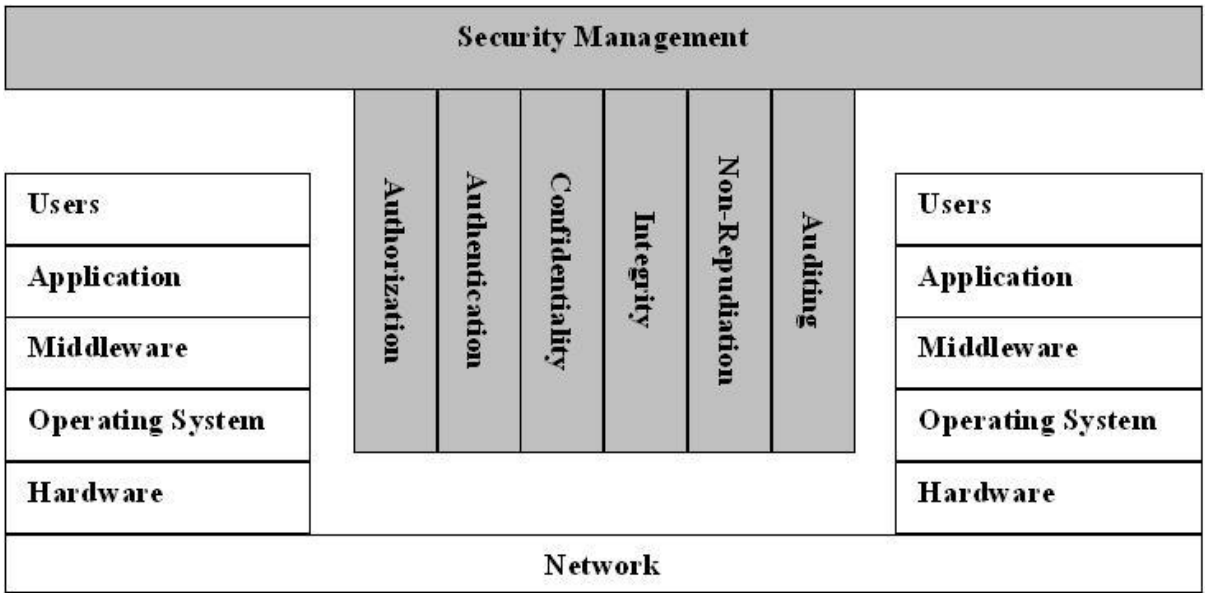

Figure 1. UT Model (adopted from [6])

Figure 2 extends the UT model to include business market segments such as finance, medical, defence and telecommunications [5]. It conveys the need to take into account specific characteristics and requirements for security services for various business segments. Hence, in the design of a secure distributed system, it is necessary not only to address the technical security requirements but also the business needs of the various applications. With respect to the UT model, the middleware we consider in this report is SOA. Over the recent years, SOA attracted considerable industry attention because of the benefits it offers such as allowing interoperability over a heterogeneous environment and integration of legacy applications, amongst others [1]. SOA can be used to build new solutions leveraging services, to integrate existing applications or to cleave apart existing applications. Although SOA offers several benefits, security is one of the main roadblocks for enterprises, delaying the development and deployment of their services [2].

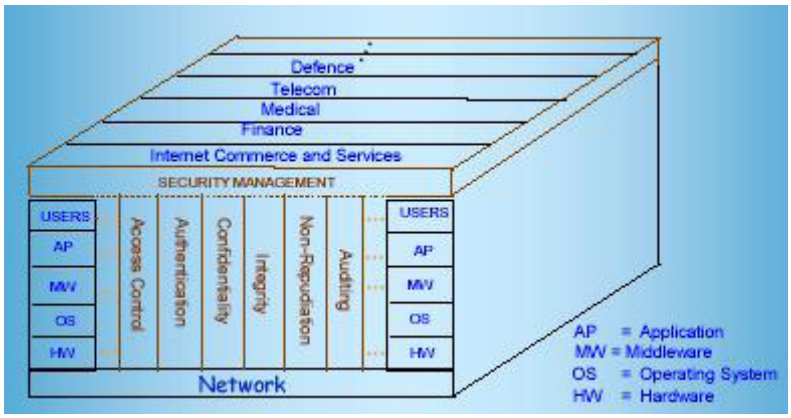

Figure 2. UT Model with vertical segments (adopted from [6])

In the next sections, we discuss some of the most common security threats faced by SOA systems. We then discuss some mitigation strategies to counter those threats. Before we move on, we would like to draw the reader's attention to the following note:

It is important to note that while SOA and Web services are usually thought to be synonymous, technically they are not. Web services technology is an important tool and *one* implementation mechanism of SOA. However, there may be other implementation mechanisms that are more suitable in any given use case. For instance, [7] claims that:

> "Web services technologies are not *necessarily* the best choice for implementing SOAs — if the necessary infrastructure and expertise are in place to use COM (Component Object Model) or CORBA (Common Object Request Broker Architecture) as the implementation technology and there is no requirement for platform neutrality, using SOAP[8]/WSDL[9][1] may not add enough benefits to justify their costs in performance."

Note: From this point onwards in this report, when we use the term SOA, we implicitly mean 'SOA implemented using the Web Services technology'. Our focus is on the Web services technology because it is the industry standard to implement SOA. Also Defence has been approved to use the Defence Operations Platform (DOP), a SOA based on the Web services technology.

4. SOA Security Vulnerabilities

As discussed in Section 3, SOA is a type of middleware. It is affected by classical security vulnerabilities affecting hardware, operating systems, and in turn any software built using the operating systems (see Figure 3). SOA[2] is also affected by Web application vulnerabilities as it is commonly built on top of (thus leveraging) the Web protocols. We also have a new class of

[1] SOAP and WSDL are Web services technologies.
[2] Please note that SOA/Web services technologies can also be used to build Web applications. We distinguish Web services from Web applications in this report in order to explicitly discuss security vulnerabilites in Web services technologies.

vulnerabilities specifically affecting an SOA, which arise due to the nature of SOA design, and new protocols and message formats supporting an SOA. The grey layers in Figure 3 correspond to SOA.

Business Processes Layer Vulnerabilities
Web Services Layer Vulnerabilities
Web Application Vulnerabilities
Classical Vulnerabilities in Hardware, Operating Systems and Software

Figure 3: Vulnerabilities affecting SOA

4.1 Classical vulnerabilities

Classical security vulnerabilities are those that can be exploited without using more recent Web technologies. An example is buffer overflows [10]. Such vulnerabilities are listed and updated in the U.S. National Vulnerability Database (NVD)[3] using a standard. The standard allows for automated vulnerability management, security measurement, and compliance. There are also other sources that list classical vulnerabilities such as The Open Source Vulnerability Database (OSVDB)[4], US-CERT Vulnerability Notes Database[5], MITRE Common Vulnerabilities and Exposure[6], and SecurityFocus[7]. Research studies on classical vulnerabilities include the RISOS study [11] (vulnerabilities in Operation Systems), the classifications by Aslam et al. [12], Krsul [13], Tsipenyuk et al. [14] and the Naval Research Laboratory (NRL) taxonomy [15]. Other sources of vulnerability classification include books written by Thompson et al. [16] and Howard et al. [17].

As SOA leverages existing operating system, software and hardware infrastructure, the security vulnerabilities listed in the sources mentioned above are in general applicable to SOA. Therefore, suitable mitigation strategies must be applied.

4.2 Web application vulnerabilities

Web application vulnerabilities occur both at the middleware and application levels (see Figure 2). The Web Application Security Consortium[8] created the Web Security Threat Classification [18] which clarifies and organises Web applications' vulnerabilities, and develops and promotes an industry standard terminology for describing those vulnerabilities. Similarly, the Open Web Applications Security Project (OWASP)[9] maintains and classifies some of the most critical Web application vulnerabilities. The Application Vulnerability Description Language (AVDL) [19], proposed by the OASIS AVDL TC, is a comprehensive

[3] http://nvd.nist.gov/
[4] http://osvdb.org/
[5] http://www.kb.cert.org/vuls/
[6] http://cve.mitre.org/
[7] http://www.securityfocus.com/
[8] http://www.webappsec.org/
[9] http://www.owasp.org/index.php/Main_Page

language based on XML that can be used to communicate about specific Web application security vulnerabilities, techniques that were used for discovering those vulnerabilities, and finally security measures to mitigate the corresponding threats.

We introduce some of the common Web application security vulnerabilities affecting SOA such as cross-site scripting (XSS) and injection attacks. As SOA leverages and is built on top of Web technologies, vulnerabilities associated with such technologies also affect SOA. Therefore, appropriate mitigation strategies must be applied.

4.2.1 Injection attack

Vulnerabilities (bugs) in software can be exploited by means of injecting code into such software in order to change the course of its execution. Code injection attacks in the case of Web applications typically involve SQL statements or scripting language code such as PHP or ASP. SQL injection attacks involve the insertion of malicious code into SQL statements to return inappropriate data, to produce an error which reveals database access information, or even worse to delete a table in the database. To counter an SQL injection attack, it is important to ensure that any SQL statements received from users are verified by means of appropriate threat-detection rules before executing those statements.

4.2.2 Cross-site scripting (XSS) attack

XSS attacks [20] are a special case of code injection. Such attacks occur when a malicious client-side script is injected into Web pages viewed by other users. XSS attacks are of three broad types:

1) Reflected or non-persistent XSS attack is the most common type. In such an attack the injected code is 'reflected' off the Web server as a response when a client sends a request such as a search string or any other input. The injected code is sent to the Web server when the client clicks on a malicious link (that includes the malicious code) on a Web page or in their email. The reflected code is executed on the client's browser as it appears to be coming from a legitimate Web server, thus causing an attack to occur.

2) Stored or persistent XSS attack occurs when the malicious code is permanently stored on an affected Web server. It can be stored in a database, in a message forum or blog, for example. When a client accesses such information sources, the malicious code is executed in their browser thus causing a XSS attack to occur.

3) DOM-based attacks are different to the other two attacks in the sense that these attacks occur by exploiting client-side code directly. The Document Object Model (DOM) [21] is used by client-side scripting languages such as JavaScript to manipulate XML content. Vulnerabilities in DOM can be exploited to manipulate the XML content when it is processed on the client-side before displaying it to the client.

One way to mitigate XSS attacks is to validate input fields from Web pages. Another approach is to disable client-side scripts from running. Some Web applications are designed to operate without the need for client-side scripts. The users of such Web applications are not susceptible

to XSS attacks when client-side scripting is disabled. However, client-side scripting enhances user experience due to its speed (being client-side, the script runs immediately without having to contact the Web server) and also reduces the load on the Web server. Therefore, not all Web applications can do without client-side scripting.

4.2.3 Cross Site Request Forgery (CSRF)

CSRF attacks [22] are the opposite of the XSS attacks. In the case of XSS attacks, a client's browser is affected because it trusts the information coming from a Web server. Whereas in the case of a CSRF attack, it is the Web server believing that the client is genuine and authenticated. Therefore, it trusts the information coming from the client and processes it. However, if the client is malicious or if it has been infected with malicious code, the Web server is susceptible to CSRF attacks. One method to mitigate CSRF attacks is having the client authenticate each time they perform a high priority operation or an operation with security implications. Another method is to use cryptographically generated random tokens that are synchronised between the client's browser and the server. If the client is infected with malicious code, a CSRF attack message generated by the malicious code will not have a valid token. This allows the Web server to reject the attack message. The OWASP CSRF guard[10] project proposes a synchronised token method to mitigate CSRF attacks.

4.2.4 Vulnerabilities in protocols and session management

Flaws used in authentication protocols may be exploited to capture authentication credentials and gain unauthorised access. If a Web site does not use appropriate transport layer security such as SSL/TLS [23], then the authentication credentials sent in plain text may be captured by rogue users/organisations and misused. Similarly, attacks can occur if Session IDs are visible to illegitimate users and if session timeouts are not properly set. Therefore, appropriate transport and application layer security protocols must be used. If any flaws are observed in those protocols, they must be immediately addressed. The lifetime of sessions must be appropriately set.

4.2.5 Security misconfiguration

If a Web site's security is not configured correctly, an attacker may exploit such a vulnerability to gain unauthorised access. For instance, if parts of a Web site are not protected by a security policy, an intruder may upload malware into that area of the Web site and potentially cause complete system compromise. Failure to restrict URL access to privileged parts of a Web site means unauthorised users may be able to access sensitive information such as configuration files and user passwords. Therefore, it is very important that Web applications make use of appropriate security management tools and well-trained security administrators who are able to configure and manage the Web applications' security appropriately.

[10] https://www.owasp.org/index.php/Category:OWASP_CSRFGuard_Project

4.2.6 Insecure cryptographic storage

If sensitive data stored at a Web site is encrypted using poor or home grown algorithms or even worse not encrypted at all, it may lead to breach of data confidentiality in case of an attack on the Web site. The use of weak hashes to protect passwords is another common flaw. Web site security administrators are recommended to employ strong encryption and hash algorithms in order to make sure such vulnerabilities do not arise.

4.2.7 Username enumeration

An attacker may use a 'username enumeration' script to query a database to determine if a supplied list of usernames is valid or not, thus enabling the attacker to experiment with different usernames. Once a valid username is found, the attacker can try widely used passwords (for test purposes) such as "test", "admin" and "password" to gain illegitimate access. Such attacks can be mitigated by displaying error messages to such username enumeration scripts that prevent disclosure of valid usernames. Also a timed block out (for one hour or more depending on the sensitivity of the Web application) may be used if a client, for instance, is unable to provide correct authentication credentials *n* times in a row[11]. Finally, making sure all the trivial username/password combinations created for testing purposes are deleted before the application is deployed online mitigates such attacks.

4.3 SOA-specific vulnerabilities

We now discuss some of the security vulnerabilities associated with the Web services and business processes layers of SOA, and their related technologies.

4.3.1 Web Services Layer

4.3.1.1 WSDL scanning

A Web service's WSDL statement advertises its operations, parameters and network bindings. Some of these (internal) operations are to be used only by the service provider, for example, administrative operations. The rest of the operations (external operations) may be invoked by any service consumer. As a Web service's end point is available in its WSDL statement, an attacker can try to guess its internal operation's name and invoke it via the endpoint. Such an attack is called WSDL scanning [24]. WSDL scanning can be mitigated by using appropriate access control mechanisms such as employing an XML firewall (more information follows in Section 6) on the internal operations or by deploying internal operations to private Web services and hosting them on internal Web servers.

4.3.1.2 Metadata spoofing

An attacker may modify Web service-related metadata such as a WSDL statement or associated WS-Security [25] policy. For instance, the Web service's endpoint may be modified for the attacker to establish a man-in-the-middle attack for eavesdropping or even worse

[11] For instance, n can be equal to 3.

modification of Web service data [24]. In order to mitigate such attacks service consumers must carefully verify the authenticity of Web service metadata. However, one must note that there are no standard mechanisms to verify metadata authenticity.

4.3.1.3 Attack obfuscation

XML Encryption [26] and XML Signature [27] standards are used to provide for encryption and digital signature services for Web service messages (such as SOAP). Such encryption technologies may be used by an attacker to conceal malicious code which executes on decryption. To mitigate attack obfuscation [24], the schema of encrypted content must be validated for safety after decryption rather than before decryption. However, this can be performance intensive as it involves XML and cryptographic processing. Therefore, stepwise decryption and validation is recommended in [24] to minimise performance impact. If an attack is found after decrypting a part of the message, the message can then be discarded as soon as the attack is found, thus avoiding processing the entire message.

4.3.1.4 Oversized cryptography

The WS-Security standard does not enforce restrictions on what parts of a security header in a SOAP message can be encrypted or even on the size of an encrypted message. This means an attacker is able to cause a denial of service by sending very large encrypted junk security headers to a Web service. Such messages cause a high load on the CPU of the Web server hosting the service as it is tries to decrypt those messages, in turn creating availability issues. Another variation on this attack is to send an oversized security header using chained encryption keys as discussed in [24]. Such an attack leads to extremely high consumption of memory and CPU. Oversize cryptography attacks can be mitigated by having the service consumer conform very strictly to the Web service's security policy (as stated using WS-Security Policy [28] standard). For instance, a service consumer's request may only be processed if the security header elements in the incoming SOAP message exactly match the security policy's schema requirements.

4.3.2 Business Processes Layer

4.3.2.1 BPEL scanning

A business process's WS-BPEL (BPEL) [29] statement may be subjected to a 'BPEL scanning' attack similar to the WSDL scanning attack described earlier. Mitigation strategies similar to the WSDL scanning attack described above may be applied.

4.3.2.2 Metadata spoofing

Metadata spoofing described earlier (for Web services) is also applicable to the business processes. For instance, an attacker may modify a business process's endpoint references in its BPEL statement. Mitigation strategies similar to those for the metadata spoofing attack described above for the Web services layer may be applied.

4.3.2.3 BPEL state deviation

A BPEL engine may have many process instances running at the same time and communication endpoints open at all times to receive incoming messages. An attacker can flood an engine on those endpoints with many BPEL messages that conform to the schema but have no meaningful content [24]. The computational resources of the BPEL engine quickly

become exhausted if such an attack happens. In order to mitigate such attacks, as few computational resources as possible should be used to reject such invalid messages. One such approach (firewall-based) is proposed in [30].

4.3.2.4 Instantiation flooding (direct and indirect)

BPEL engines instantiate a new process when they receive an incoming 'receive' message. When a receive message is received a BPEL engine pauses its current execution. It continues execution only after the incoming message is fully received. An attacker may exploit this feature of BPEL engines by repeatedly sending invalid receive messages. Such messages severely affect the BPEL engine by decreasing or even nullifying its availability to legitimate messages. Indirect flooding of BPEL engines is also possible when the attack target differs. For instance, an attacker can use an instantiated process on one BPEL engine to cause a flooding attack on another BPEL engine. This is possible if the attacker invokes a business process on the first BPEL engine that interacts with another Web service or business process on the second BPEL engine. Several variations of instantiation flooding attacks are discussed in [24]. Protecting BPEL engines against flooding attacks is not trivial as in many cases incoming messages need to be semantically analysed before discarding invalid messages. Also the semantics of a business process are not disclosed in its process description (BPEL statement).

4.3.2.5 WS-Addressing spoofing

The WS-Addressing [31] specification discusses how to address Web service and business process endpoints in a standard way. An attacker can modify endpoint references used in WS-Addressing headers to point a BPEL engine to invalid or malicious services or processes [24]. This attack can be compounded if used as a flooding attack maximising the workload produced by each attack message on a BPEL engine. Address spoofing can be mitigated by verifying the validity of endpoint URLs before the process is executed by the BPEL engine. Please note that this attack also applies to the Web services layer of SOA.

4.3.2.6 Workflow engine hijacking

In this attack, address spoofing is used once again to flood a target system [24]. The difference is that the attacker's endpoint is a legitimate service (the target of the attack). The problem with identifying this attack is that sometimes the messages may even be genuine. For example, a credit card payment service used by an online gift shop may be flooded with messages at the time of the Boxing Day sale every year; the messages maybe legitimate (genuine purchases) and not an attack on the service. This means the target system tries hard to process the attack messages (in turn breaking down) and its legitimate users suffer a denial of service. The attacker leverages the processing power of the source system to tear down the target system. Workflow engine hijacking can be mitigated by verifying the validity of endpoint URLs before the process is executed by the BPEL engine.

4.3.3 SOA-specific technology vulnerabilities impacting all layers of SOA

4.3.3.1 SOAP vulnerabilities

4.3.3.1.1 Harmful SOAP attachments

SOAP messages may contain attachments of arbitrary size. An attacker may attach a virus to a SOAP message and send it for processing to the target system. Or the attacker may send very

large encrypted attachments that are of absolutely no value to the target system. Such attachments are difficult to process and may cause denial of service. To mitigate such attacks, SOAP attachments may be blocked in some cases. In other cases where attachments are necessary, they may either be filtered based on a MIME-type or passed through a firewall or an anti-virus solution [32].

4.3.3.1.2 SOAPAction spoofing

In this attack, an attacker modifies the 'SOAPAction' element in a HTTP header in order to execute an action that is not predefined by the Web service provider. Such attacks can either be executed by the client directly or by an intruder (man in the middle). Examples of such attacks are discussed in [24]. SOAPAction spoofing attacks can be mitigated by strictly verifying if the action in the SOAP body matches the action in the HTTP header [24]. If they do not match, the incoming message should be rejected.

4.3.3.2 XML vulnerabilities

4.3.3.2.1 XML External Entity (XEE) attack

The XML specification allows for external data to be included into XML using URIs dynamically at the time of processing XML [32]. An attacker can take advantage of this fact and replace the data or information being collected from the URI with malicious code. In order to mitigate XEE attacks, an XML parser should strictly prohibit all external entities in untrusted XML.

4.3.3.2.2 XPath injection

The XPath [33] specification is used to navigate the content of an XML document. An Xpath injection attack (similar to SQL injection attack) is possible and can be used to inject an XPath expression and access unauthorised information from an XML database. Such an attack can be mitigated by making sure that an XPath expression itself does not contain another XPath expression [32].

4.3.3.2.3 XML Denial of Service (DoS) attacks

DoS attacks are caused due to resource exhaustion (processor, memory, or network bandwidth) at the server hosting a Web service or a business process. We discuss some of the DoS attacks associated with XML here.

XML consumes large amounts of memory for it to be parsed and processed. The Document Object Model (DOM) [21] approach for parsing and processing XML consumes a very large amount of memory. This is because an in-memory object representation of the entire XML document, that requires much more memory space than the XML document itself, is required. In [24], the authors observed a rise in memory consumption by a factor of 2 to 30 in commonly used Web service frameworks when using DOM over another model for XML parsing — the Simple API for XML (SAX) [34]. Therefore, an oversize XML payload in a SOAP message can be used to easily cause a DoS attack. One way to counter an oversize payload attack is to limit the size of incoming SOAP messages.

An attacker can exploit certain features of XML to cause DoS using "coercive parsing" [35]. For instance, XML can become verbose and complex in parsing when using namespaces. An

example attack is discussed in [24]. Coercive parsing attacks based on complex or deeply nested documents can be mitigated using schema validation against the service's WSDL before fully parsing the incoming message. However, attacks based on misusing namespace declarations are harder to prevent as the XML specification does not place restrictions on the way namespaces are defined.

An attacker can exploit a feature of Document Type Definitions (DTDs) that enables them to pull in entities which are defined in a DTD. By pulling in entities recursively, an attacker can create an XML message which explodes in memory and causes a DoS attack [32]. Such an attack can be mitigated by limiting the size of XML payloads in incoming SOAP messages.

A useful feature of XML is nesting of elements. However, this feature can be used to stress and break an XML parser by sending a SOAP message with XML content that is for instance a thousand or more elements deep, thus causing a DoS attack [35]. Such an attack can be mitigated by limiting the size of incoming SOAP messages.

Completely valid XML messages can be used to cause a DoS attack called "replay attack" in [35]. An attacker can send repetitive SOAP messages carrying XML payloads that are valid and the requests are well formed. However, the intention to send those messages is malicious (to cause a DoS attack) and attacker does not have a business reason to send those messages. Such attacks can be thwarted by using unique session tokens in SOAP messages such as nonces (unique numbers used only once).

4.3.3.2.4 Schema poisoning

Schema poisoning involves modifying the XML schema information to attack a target system. An attacker may intercept an XML schema before it reaches a client (from a server) and modify it. Schema poisoning may cause a DoS attack as the XML parser may hang or reach an inconsistent state as it does not have relevant schema information for parsing the XML document. Minor modifications of the schema (such as modifying data types) may mean an inaccurate response is sent to the client. Schema poisoning attacks can be thwarted by protecting XML schemas against unauthorised modification [36].

Many of the threats described in this section can be mitigated by making use of suitable authentication, confidentiality, integrity, and authorisation standards such as Security Assertion Markup Language (SAML) [37], WS-Security [25] and eXtensible Access Control Markup Language (XACML) [38], and Commercial of the Shelf (COTS) solutions such as IBM Tivoli Identity Manager (TIM) [39], IBM Tivoli Access Manager (TAM) [40] and CA SOA Security Manager [41] for authentication and for authorisation. Machines/non-human users should be clearly identified and authenticated by the identity provision and authentication services.

Although IBM TIM (part of the Defence Operations Platform (DOP)) offers an authentication service, it does not offer federated authentication to external service providers. Similarly, IBM TAM (part of the DOP) does not offer a distributed authorisation service. For example, when a composite Web service is invoked, an external service provider's service may be invoked. In

such a scenario, a device (the DataPower[12] appliance) authenticates itself to the external service provider in order to gain access to the service. However, in many such cases, the external service provider requires end-user authorisation, and not just the device authorisation. As far as we know, this end-user authorisation to external service providers is not currently possible within the DOP. Indrakanti et al. [42-44] proposed a comprehensive authorisation framework for SOA, which offers end-user authorisation to external service providers. The authorisation framework also addresses other design requirements proposed in [45-47].

Suitable auditing and non-repudiation services provided by COTS tools such as IBM Tivoli Compliance Insight Manager [48] can be leveraged. Message level security (confidentiality and integrity) must be provisioned by using an appropriate gateway (such as CISCO ACE XML Gateway [49] or Sentry XML Gateway [50]) and/or middleware product such as an ESB implementing suitable standards such as WS-Security, XML Encryption [26] and XML Signature [27]. Also SOA-specific technology threats (such as XML threats and SOAP threats) must be mitigated using appropriate tools and solutions described in Section 6.

In Sections 5 and 6, we describe some suitable SOA threat mitigation strategies by means of SOA security standards and their implementation in COTS solutions.

5. SOA Security Standards

There have been several efforts striving to provide SOA security standards for authentication, confidentiality and integrity amongst others. We introduce some of the relevant SOA security standards here.

XML Signature [27] — XML Signature is a foundational technology for the WS-Security [25] specification and for Web services security in general. It is core to the WS-Security, XML Key Management Specification (XKMS) [51] and other Web services security standards. It is also useful for transporting shared secret keys that are needed by XML Encryption [26]. XML Signature enables the encoding of digital signatures into XML. The XML Signature Syntax and Processing W3C Recommendation defines the XML signature syntax and associated processing rules.

XML Encryption [26] — Similar to XML Signature, XML Encryption is built using shared-key encryption technology, and is a W3C recommendation. The core requirements for XML Encryption are that it must be able to encrypt an arbitrarily sized XML message and it must do so efficiently. The reason XML Encryption is required over and above transport-level encryption such as SSL is that message confidentiality should be maintained when a message takes multiple hops on its way to its destination. This will be common when shared services are used. XML Encryption also preserves message confidentiality when an XML message is

[12] http://www-01.ibm.com/software/integration/datapower/xs40/

stored even after it reaches its final destination. XML Encryption applies standard algorithms to data and then stores that encrypted result in XML.

XML Key Management Specification [51] — The XML Key Management Specification (XKMS) is built on top of and complements the XML standards for digital signature and encryption. XML signature and XML encryption technologies scale best when they use public key cryptography. Public key cryptography requires a supporting Public Key Infrastructure (PKI) [52, pp.62-110] to handle distribution, certification and life-cycle management (for example, the revocation) of keys. Web services themselves provide a different approach by enabling the PKI to be accessed as a service; hence, there is no need for each Web service requestor and provider to build their own PKI. XKMS aims to do just that. It specifies protocols for distributing and registering public keys suitable for use in conjunction with the XML Signature and the XML Encryption standards.

SAML — Security Assertion Markup Language (SAML) [37] proposed by the OASIS Security Services Technical Committee, is an XML-based security specification for exchanging authentication and authorisation information about a user or subject. It defines an XML schema and definition for security assertions. The assertions are of three types — authentication, any security related attributes for the subject, and the authorisation decisions given based on the security and privilege attributes. SAML can also be extended to send any arbitrary security assertions. This shows that SAML is primarily used to carry security and privilege attributes related to a subject from one entity to another entity in a network. The only requirement is that both the entities comply with the SAML standard.

XACML — EXtensible Access Control Markup Language (XACML) [38] is an XML-based policy language for access control that has been standardised by OASIS. XACML describes both an access control policy language and leverages an SAML profile for XACML for request/response messages. The SAML-profile for XACML can be used to express queries about whether a particular access should be allowed (requests), and also to describe answers to those queries (responses). The XACML policy language is heavily based on the XACL [53] policy language, but uses more generic terminology. An XACML policy specification is composed of one or more *rule*, *policy*, and *policy set* elements. A rule is the most elementary unit of policy. Rules must be encapsulated in a policy for them to be exchanged between entities. A rule is composed of a *target*, an *effect*, and a *condition*.

5.1 WS-Security and related standards

In April 2002, IBM and Microsoft published a joint whitepaper called 'Security in a Web Services World: A Proposed Architecture and Roadmap' [54]. This whitepaper describes a set of security standards and technologies meant to create a unifying approach for dealing with security in a Web services world. Just as WS-Security allows security mechanisms such as Public Key Infrastructure (PKI) and SAML to participate in Web services security, the Web Services Architecture Roadmap generalises many of the security functions that previously existed in other domains and proposes a framework for meeting the security requirements of the Web services domain. The proponents of this framework and the standards bodies they are working through are accomplishing this by first rolling out foundational specifications

such as WS-Security (which, in turn, was built on XML Signature, XML Encryption, SAML and other security-token standards) and then developing other standards that rely on these foundational standards.

WS-Security [25] — Describes extensions to SOAP for secure messaging. It is a general-purpose mechanism for associating security tokens with SOAP messages. WS-Security builds on and is fully compatible with established and mature security technologies such as SSL, IPSec[13], XML Signature and XML Encryption. It is designed to address message integrity, message confidentiality, message authentication and the encoding of security tokens that travel with the messages being secured.

WS-Policy [55] — Defines how to express the capabilities and constraints of security policy. WS-Policy allows organisations exposing Web services to specify the requirements of their Web services for issues such as privacy or security. The WS-Policy is a high-level specification providing the basic constructs required to compose a particular policy language (such as WS-SecurityPolicy [28]). Closely related to the WS-Policy specification is the WS-PolicyAssertions [56] specification, which provides some basic policy assertions that would apply to any type of policy, and the WS-PolicyAttachment [57] specification, which gives guidance on how to attach a policy to a resource. The WS-SecurityPolicy is a specific type of policy using the WS-Policy framework that answers certain security requirement and configuration questions for a Web service, such as the types of encryption algorithms that are to be supported, the parameters that are to be encrypted and the types of security tokens that are to be specified.

WS-Trust [58] — Describes the model for establishing both direct and brokered trust relationships including intermediaries. The Web Services Trust Language (WS-Trust) uses the secure messaging mechanisms of WS-Security to define additional primitives and extensions for the issuance, exchange and validation of security tokens. WS-Trust also enables the issuance and dissemination of credentials within different trust domains. To secure a communication between two parties, the two parties must exchange security credentials (either directly or indirectly). However, each party needs to determine if they can 'trust' the asserted credentials of the other party. This specification defines extensions to WS-Security for issuing and exchanging security tokens and ways to establish and access the presence of trust relationships. Using these extensions, applications can engage in secure communication designed to work with the general Web services architecture.

WS-Privacy — This specification will enable users to state privacy preferences and Web services to state and implement privacy practices. At the time of writing this report, there is no WS-Privacy standard available. The Platform for Privacy Preferences (P3P) [59] project defines a protocol that allows websites to specify how they intend to use private information of users. The protocol allows users to make an informed decision with respect to privacy; whether or not to use a website. The P3P standard maybe leveraged and perhaps enhanced to specify and enforce the privacy preferences of Web services' users.

WS-SecureConversation [60] — Describes how to manage and authenticate message exchanges between parties, including exchanging security contexts and establishing and

[13] Internet Protocol Security. Provides end-to-end security in the internet layer of the TCP/IP suite.

deriving session keys. The Web Services Secure Conversation Language (WS-SecureConversation) is built on top of the WS-Security and WS-Policy models to provide secure communication between Web services.

WS-Federation [61] — Describes how to manage and broker the trust relationships in a heterogeneous federated environment, including support for federated identities. The Web Services Federation specification defines mechanisms to allow different security realms to federate by allowing and brokering trust of identities, attributes and the authentication between participating Web services. The mechanisms defined in the specification can be used by both passive and active requestors. The Web service requestors are assumed to understand the new security mechanisms and be capable of interacting with Web service providers.

WS-Authorisation — This specification will define how Web services' authorisation data and policies are managed. At the time of writing this report, there is no WS-Authorisation standard available. However, several authorisation models have been proposed for Web service authorisation [42-44, 62-68]. The authorisation framework proposed by Indrakanti et al. [42-44] provides the features [45-47] required for a comprehensive SOA authorisation framework.

WS-Auditing — A standard for auditing does not exist yet for Web services. After a service has been invoked, there is no way to determine who has used the service and from where the request originated. As a result, no audit trail exists that can be used later to investigate possible breaches in security; there is no way to determine who has done what and at what time. However, a COTS solution such as IBM Tivoli Compliance Insight Manager [48] may be relied upon for auditing Web services transactions.

6. SOA Security Products – XML Appliances and SOA Solutions

In this section, we briefly introduce some of the commonly used COTS XML appliances and SOA security products. Each sub-section introduces a particular vendor's product(s).

6.1 Layer7

SecureSpan XML Firewall and VPN; Mainframe SOA Gateway

Layer 7 XML Firewall [69] and SOA Gateway [70] products support authentication and authorisation services using related standards such as SAML, XACML and WS-Security. They also support single-sign-on (SSO) and federation of identities across multiple domains. They provide for message level security and support elliptic curve cryptography (according to NSA's Suite B algorithms). They support FIPS 140-2 [71] in hardware (Level 3) and software (Level 1). Support for auditing by logging message transaction-level information is provided. A variety of XML threats are mitigated.

Layer 7 products also facilitate compliance to service level agreements (SLAs), interoperability with other SOA solutions, SOA policy management and provide deployment flexibility over various hardware-software combinations.

6.2 IBM

WebSphere DataPower XML Security Gateway XS40; Tivoli Suite

IBM's XML Gateway XS40 [72] protects against a variety of XML threats. It provides message level security — encryption/decryption and signing/verification of entire messages or parts (individual XML fields) of XML messages. It also provides authentication and fine-grained authorisation for Web services leveraging standards such as SAML, XACML and WS-Security.

XS40 can be integrated with Tivoli Identity Manager [39] and Tivoli Access Manager [40] to manage user identities and control access to an enterprise's applications. Tivoli Federated Identity Manager [73] can be used with the XS40 to provide policy-based integrated security management for federated Web services in an SOA environment. Tivoli Compliance Insight Manager [48] and Tivoli zSecure [74] provide auditing services.

6.3 Intel

Intel SOA Expressway

Security Gateway is part of the Intel SOA Expressway [75]. It supports message level security (XML encryption and XML signature). It also supports hardware-based private key storage and provides support for encryption algorithms including DES, 3DES and AES.

SOA Expressway supports the WS-Security standard, X.509 Tokens, HTTP Basic Authentication, Username/Password Tokens, X.509 path validation and Certificate Revocation List (CRL). It has support for SSL/TLS origination and termination as well as support for an optional cryptographic accelerator for digital signature, encryption and security token processing. It also offers protection against a range of XML threats. It also supports centralised security policy management. It provides an onboard digital forensics (auditing) capability.

6.4 Cisco

Cisco ACE XML Gateway

Cisco ACE XML Gateway [49] supports authentication and authorisation services using related standards such as SAML, XACML and WS-Security. It supports encryption and digital signatures for information confidentiality and integrity. It provides native integration with business directory and identity systems such as Lightweight Directory Access Protocol (LDAP), Kerberos, Microsoft Active Directory and IBM Tivoli Access Manager. It protects

against Secure Sockets Layer (SSL) key hijacking by persistently storing SSL private keys in the hardware platform.

Cisco ACE XML Gateway supports centralised policy management and decentralised enforcement. It provides logging, auditing and monitoring capabilities. It meets with audit and compliance requirements and provides non-repudiation capabilities. It supports traffic and service-level agreement (SLA) monitoring and reporting. It also defends against a range of XML threats.

6.5 Forum

Sentry XML Gateway Appliance / Software

Sentry XML Gateway is sold either as a hardware appliance or software. The feature set is the same for both the hardware and software [50] with the exception being FIPS hardened key storage and hardware based cryptographic acceleration for signature, encryption, and SSL, which is available only on the hardware appliance. The appliance provides for optional FIPS Level 3 Hardware Security Module (HSM). Sentry XML Gateway supports authentication using related standards such as SAML and WS-Security. It only provides for network and WSDL message level access control. It supports message level security (XML encryption and XML signature). Sentry XML Gateway supports policy management, and can interface with Tivoli Access Manager. It provides for XML security and mitigates various types of XML threats.

6.6 Vordel

Vordel XML Gateway

Vordel XML Gateway supports authentication and authorisation for Web services using standards such as SAML, XKMS, WS-Security, and XACML. The gateway supports message level confidentiality and integrity by supporting XML Encryption and XML Signature standards. The gateway supports identity management services by integrating with many commercially available identity management products such as Oracle Access Manager, IBM Tivoli Access Manager and Microsoft Active Directory. It mitigates many of the XML, SOAP and WSDL threats discussed in Section 4. The product datasheet [76] lists those threats which are mitigated by the gateway. The gateway also supports auditing by creating logs that are tamper-proof for all transactions.

6.7 CA

CA SOA Security Manager

The CA SOA Security Manager [41] supports authentication, authorisation, single-sign-on, and identity federation via standards such as WS-Security and SAML. It supports message

level confidentiality and integrity via the XML Encryption and XML Signature standards. The CA SOA Security Manager supports policy management and enforcement for Web services. It supports auditing and reporting by logging security-related events for all transactions. It also prevents a range of XML threats.

7. Conclusion

SOA provides many benefits such as cost savings to organisations. Computer systems and distributed systems in particular face several security risks. They are vulnerable to both active and passive attacks. A distributed system is composed of several layers including a fully functional network, with nodes in that network running on a piece of hardware. Operating systems and other software such as middleware are deployed on the hardware in turn enabling running of multiple applications. SOA is a type of middleware in a distributed system, and is therefore vulnerable to security risks affecting each of the layers it is composed of and built upon. Security services such as confidentiality, integrity, authentication, access control/ authorisation, non-repudiation and auditing are used to mitigate such security risks.

SOA is affected by classical system vulnerabilities and Web application vulnerabilities as it is built upon and leverages classical and Web application technologies. In this report, we described common SOA-specific security vulnerabilities posing threats to SOA systems. Several security standards have been proposed by organisations such as OASIS and W3C to design and implement security services for SOAs. We briefly introduced those standards in this report. We also introduced some of the commonly used COTS SOA security products.

SOA security standards implemented in COTS solutions can be used to mitigate the security threats faced by SOA systems. However, not all security standards are available. The WS-Authorisation and the WS-Privacy standards are yet to be proposed. Several authorisation models have been proposed for Web service authorisation [42-44, 62-68]. The authorisation framework proposed by Indrakanti et al. [42-44] provides the features [46, 47] required for a comprehensive authorisation framework for SOA. Similarly, the P3P standard [59] maybe leveraged and perhaps enhanced to specify and enforce the privacy preferences of Web services' users.

8. References

1. Yoon, T. and Carter, P. (2007) Investigating the Antecedents and Benefits of SOA Implementation: A Multi-Case Study Approach. In: *AMCIS 2007,* Keystone, Colorado, USA
2. CA Advisor (2009) CA Survey Finds Security Concerns Slow SOA/Web Service Implementation. *Security Management Newsletter*
3. Neumann, B. C. and Ts'o, T. (1994) Kerberos: An Authentication Service for Computer Networks. *IEEE Communication Magazine* **32** (9) 33-38
4. ITU-T Recommendation (June 1997) *Information Technology - Open Systems Interconnection - The Directory: Authentication Framework*.
5. Hewlett Packard (1997) *Authorization Server: Administration and Programming Reference Guide*
6. Indrakanti, S. (2007) *PhD Thesis: On Engineering Authorization Systems for Web Services based Service-Oriented Architecture*. Sydney, Macquarie University
7. D. Booth, et al. (2004) *Web Services Architecture,* http://www.w3.org/TR/ws-arch/.
8. World Wide Web Consortium (W3C) (2003) *SOAP v1.2,* http://www.w3.org/TR/soap12-part1/.
9. World Wide Web Consortium (W3C) (2004) *Web Services Description Language (WSDL) v2.0,* http://www.w3.org/TR/wsdl.
10. Meunier, P. (2006) Software Development and Quality Assurance In: Bidgoli, H. (ed.) *Handbook of Information Security*. Vol. 2.
11. R. P. Abbott, et al. (1976) *Security Analysis and Enhancements of Computer Operating Systems*. 76-1041, Washington, DC 20234, Institute for Computer Sciences and Technology, NATIONAL BUREAU OF STANDARDS
12. T. Aslam, I. Krsul and E.H. Spafford (1996) Use of a taxonomy of security faults. In: *19th National Information Systems Security Conference,* Baltimore, MD
13. Krsul, I. V. (1998) *Software vulnerability analysis: PhD dissertation*. Purdue University
14. K. Tsipenyuk, B. Chess and G. McGraw (2005) Seven pernicious kingdoms: A taxonomy of software security errors. *IEEE Security and Privacy* **3** (6) 81-84
15. C. Landwehr, et al. (1994) A taxonomy of computer program security flaws. *Computing Surveys* **3** (26) 211-254
16. H.H. Thompson and Chase, S. G. (2005) *The Software Vulnerability Guide*, Charles River Media
17. M. Howard, D. LeBlanc and J. Viega (2005) *19 Deadly Sins of Software Security*, McGraw Hill Osborne Media
18. Web Application Security Consortium. *Web Security Threat Classification*. [Accessed 13 January, 2011]; Available from: https://files.pbworks.com/download/p5LJksUNog/webappsec/13247059/WASC-TC-v2_0.pdf.
19. OASIS (2004) *Application Vulnerability Description Language (AVDL) v1.0,* http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=avdl.
20. The Open Web Application Security Project (OWASP). *Cross-site Scripting (XSS)*. (2011) [Accessed 08/08/2012]; Available from: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS).

21. World Wide Web Consortium (W3C) *Document Object Model (DOM)*, http://www.w3.org/DOM/.
22. The Open Web Application Security Project (OWASP). *Cross-Site Request Forgery (CSRF)*. (2010) [Accessed 08/08/2012]; Available from: https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF).
23. Rescorla, E. (2001) *SSL and TLS: Designing and Building Secure Systems*, Addison Wesley
24. M. Jensen, et al. (2007) SOA and Web Services: New Technologies, New Standards - New Attacks. In: *Fifth European Conference on Web Services (ECOWS)*, Halle (Saale), Germany
25. OASIS (2006) *Web Services Security (WS-Security) Specification*, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss.
26. World Wide Web Consortium (W3C) (2002) *XML Encryption Syntax and Processing*, http://www.w3.org/TR/xmlenc-core.
27. World Wide Web Consortium (W3C) (2008) *XML-Signature Syntax and Processing*, http://www.w3.org/TR/xmldsig-core/.
28. OASIS (2007) *Web Services Security Policy Language (WS-SecurityPolicy) v1.2*, http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.html.
29. OASIS (2007) *Web Services Business Process Execution Language Version 2.0*, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.
30. N. Gruschka, M. Jensen and N. Luttenberger (2007) A Stateful Web Service Firewall for BPEL In: *IEEE International Conference on Web Services (ICWS)*,
31. World Wide Web Consortium (W3C) (2004) *Web Services Addressing (WS-Addressing)*, http://www.w3.org/Submission/ws-addressing/.
32. O'Neill, M. *SOA Security: The Basics*. [Accessed 21 January, 2011]; Available from: http://www.csoonline.com/article/484120/soa-security-the-basics?page=3.
33. World Wide Web Consortium (W3C) (2005) *XML Path Language Version 2.0*, http://www.w3.org/TR/2005/WD-xpath20-20050404/.
34. SAX 2.0.1 (2004) *Simple API for XML (SAX)*, http://www.saxproject.org/.
35. Lindstrom, P. (2004) *Attacking and Defending Web Services: A Spire Research Report*.
36. *Common Attack Pattern Enumeration and Classification*. [Accessed 25 January, 2011]; Available from: http://capec.mitre.org/data/definitions/146.html.
37. OASIS (2005) *Security Assertion Markup Language v2.0*, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security.
38. OASIS (2005) *eXtensible Access Control Markup Language (XACML) Specification Version 2.0*.
39. IBM. *Tivoli Identity Manager Product Documentation*. (2010) [Accessed 20 September, 2010]; Available from: http://www-01.ibm.com/software/tivoli/products/identity-mgr/.
40. IBM. *Tivoli Access Manager Product Documentation*. (2010) [Accessed 20 September, 2010]; Available from: http://www-01.ibm.com/software/tivoli/products/access-mgr-e-bus/
41. CA. *CA SOA Security Manager*. [Accessed 31 January, 2011]; Available from: http://www.ca.com/files/productbriefs/soa_sm_pb.pdf.
42. Indrakanti, S. and Varadharajan, V. (2005) An Authorization Architecture for Web Services. In: *19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, Storrs, Connecticut, USA, Springer LNCS

43. Indrakanti, S. and Varadharajan, V. (2011) Coordination based Distributed Authorization for Business Processes in Service Oriented Architectures. In: *The Sixth International Conference on Internet and Web Applications and Services,* St. Maarten, The Netherlands Antilles
44. Indrakanti, S., Varadharajan, V. and Agarwal, R. (2007) On the design, implementation and application of an authorisation architecture for web services. *International Journal of Information and Computer Security* **1** (1/2)
45. Indrakanti, S. (2012) *On the Design Requirements for a Comprehensive SOA Authorisation Framework; DSTO-CR-2011-0251* DSTO
46. Indrakanti, S., Varadharajan, V. and Hitchens, M. (2005) Principles for the Design of Authorization Framework for the Service Oriented Architecture. In: *International Conference on Internet Technologies and Applications (ITA 05),* Wrexham, North Wales, UK: September 7-9
47. Indrakanti, S., Varadharajan, V. and Hitchens, M. (2005) Analysis of Existing Authorization Models and Requirements for Design of Authorization Framework for the Service Oriented Architecture. In: *The 2005 International Symposium of Web Services and Applications,* Las Vegas, USA: June 27-30
48. IBM. *Tivoli Compliance Insight Manager Product Documentation*. (2010) [Accessed 20 September, 2010]; Available from: http://www-01.ibm.com/software/tivoli/products/compliance-insight-mgr/.
49. CISCO. *ACE XML Gateway*. [Accessed 28 January, 2011]; Available from: http://www.cisco.com/en/US/products/ps7314/index.html.
50. Forum. *Sentry XML Gateway Appliance / Software*. [Accessed 28 January, 2011]; Available from: http://www.forumsys.com/products/soagateway.php.
51. World Wide Web Consortium (W3C) (2005) *XML Key Management Specification (XKMS 2.0), http://www.w3.org/TR/xkms2/*.
52. Stallings, W. (2000) *Network Security Essentials - Applications and Standards*. First reprint, 2001 ed, Pearson Education Asia
53. Kudo, M. and Hada, S. (2000) XML Document Security based on Provisional Authorization. In: *ACM Conference on Computer and Communications Security (CCS),* Greece: November
54. IBM Corporation and Microsoft Corporation (2002) *Security in a Web Services World: A Proposed Architecture and Roadmap, http://www.ibm.com/developerworks/library/specification/ws-secmap/*.
55. World Wide Web Consortium (W3C) (2007) *Web Services Policy Framework (WS-Policy), http://www.w3.org/TR/ws-policy/*.
56. OASIS (2006) *Web Services Reliable Messaging Policy Assertion (WS-RM Policy), http://docs.oasis-open.org/ws-rx/wsrmp/200608/wsrmp-1.1-spec-cd-04.html*.
57. World Wide Web Consortium (W3C) (2007) *Web Services Policy Attachment (WS-PolicyAttachment), http://www.w3.org/TR/ws-policy-attach/*.
58. OASIS (2005) *Web Services Trust Language (WS-Trust), http://www-106.ibm.com/developerworks/library/specification/ws-trust/*.
59. World Wide Web Consortium (W3C) (2006) *The Platform for Privacy Preferences 1.1 (P3P1.1) Specification, http://www.w3.org/TR/P3P11/*.
60. OASIS (2007) *Web Services Secure Conversation Language (WS-SecureConversation), http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.html*.

61. OASIS (2010) *WS-Federation,* http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsfed.
62. Agarwal, S., Sprick, B. and Wortmann, S. (2004) Credential Based Access Control for Semantic Web Services. *American Association for Artificial Intelligence*
63. Bertino, E., Crampton, J. and Paci, F. (2006) Access Control and Authorization Constraints for WS-BPEL. In: *International Conference on Web Services (ICWS)*: 18-22 Sept. 2006
64. Karp, A. H. (2006) Authorization-Based Access Control for the Service Oriented Architecture. In: *Fourth International Conference on Creating, Connecting, and Collaborating through Computing,* Berkeley, CA, USA: 26-27 January 2006
65. Koshutanski, H. and Massacci, F. (2002) *An Access Control System for Business Processes for Web Services*. DIT-02-102, [Technical Report] Informatica e Telecomunicazioni, University of Trento
66. Kraft, R. (2002) Designing a Distributed Access Control Processor for Network Services on the Web. In: *ACM Workshop on XML Security,* Fairfax, VA, USA: November 22
67. Mont, M. C., Baldwin, A. and Pato, J. (2003) *Secure Hardware-based Distributed Authorisation Underpinning a Web Service Framework*. HPL-2003-144,
68. Yagüe, M. I. and Troya, J. M. (2002) *Euroweb 2002 Conference. The Web and the GRID: from e-science to e-business*, Oxford, UK
69. Layer7 Technologies. *XML Firewall*. [Accessed 28 January, 2011]; Available from: http://www.layer7tech.com/products/xml-firewall.
70. Layer7 Technologies. *SOA Gateway*. [Accessed 28 January, 2011]; Available from: http://www.layer7tech.com/products/soa-gateway.
71. National Institute of Standards and Technology (Information Technology Laboratory) *FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION: SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC MODULES (FIPS PUB 140-2)*
72. IBM. *WebSphere DataPower XML Security Gateway XS40*. [Accessed 28 January, 2011]; Available from: http://www-01.ibm.com/software/integration/datapower/xs40/.
73. IBM. *Tivoli Federated Identity Manager Product Documentation*. [Accessed 28 January, 2011]; Available from: http://www-01.ibm.com/software/tivoli/products/federated-identity-mgr/.
74. IBM *Tivoli zSecure Suite, Product Documentation,* http://www-01.ibm.com/software/tivoli/products/zsecure/.
75. Intel. *Intel SOA Expressway*. [Accessed 28 January, 2011]; Available from: http://software.intel.com/en-us/articles/Intel-soa-expressway-service-gateway-home/.
76. Vordel. *Vordel XML Gateway*. [Accessed 31 January, 2011]; Available from: http://www.vordel.com/downloads/v6_gateway.pdf.

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
DOCUMENT CONTROL DATA

1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)

2. TITLE

Service Oriented Architecture Security Risks and their Mitigation

3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)

Document	(U)
Title	(U)
Abstract	(U)

4. AUTHOR(S)

Sarath Indrakanti

5. CORPORATE AUTHOR

Defence Science and Technology Organisation
PO Box 1500
Edinburgh South Australia 5111 Australia

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO-TN-1126	AR-015-420	Technical Note	October 2012

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. NO. OF PAGES	12. NO. OF REFERENCES
2012/1000290/1	07/012	Stephen Bowman, CIOG	23	76

DSTO Publications Repository

http://dspace.dsto.defence.gov.au/

14. RELEASE AUTHORITY

Chief, Command, Control, Communications and Intelligence Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for public release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SA 5111

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS Yes

18. DSTO RESEARCH LIBRARY THESAURUS

SOA, Web services, business processes, security, distributed systems, architecture, risk mitigation

19. ABSTRACT

Service Oriented Architecture (SOA) is an architectural paradigm and its aim is to achieve a loose coupling amongst interacting distributed systems. SOA is used by enterprises to efficiently and cost-effectively integrate heterogeneous systems. However, SOA is affected by several security vulnerabilities, thus affecting the speed of its deployment in organisations. In this report, we describe some of the security threats faced by SOA systems and corresponding risk mitigation measures by means of security technology, standards and products.