Service Oriented Architecture Security Vulnerabilities - Web Services

A Service-Oriented Architecture (SOA) presents new flexibility and capability for application developers due largely to the fact that SOA components are loosely coupled and exposed as independent services on a network. Therefore, application developers can access SOA services in a standardized way and without understanding how the service is implemented. This evolutionary concept in software architecture is popular because SOA reduces development time, promotes software reuse, and reduces project costs.

SOA implementations exist in almost all industries. Developers tout the unprecedented interoperability that exists between applications built with SOA. However, sometimes developers fail to secure SOA services and architectures. When combined with protocols allowed through security firewalls such as SOAP over port 80, as is common in Web Service environments, SOA can be a security disaster waiting to happen. In practice, inaccurate vendor implementations, configuration problems, and coding mistakes can lead to exploitable vulnerabilities in web services. Developers and System Administrators must understand the risk posed by these vulnerabilities and consider mitigations prior to deployment.

Some of the critical vulnerabilities that can be introduced by Web Services are listed below. The vulnerabilities are separated into two groups. The first group contains vulnerabilities specific to Web Services whereas the second group contains vulnerabilities that are more general, but still important to Web Services security.

Injection Flaws – Injection flaws occur when software does not properly validate input. An attacker could craft malicious input that causes the Web Service software to perform operations on behalf of the attacker. Classes of injection flaws include Cross Site Scripting, SQL Injection, and XPath Injection.
Mitigations - Developers must validate all web service parameters at the server prior to using them and prior to generating output. The developer should not assume clients will generate valid input or adhere to the Web Service Description Language (WSDL) specifications. Web Service gateways are another possible mitigation that can detect these types of attacks.

SNAC DoD,9800 Savage Rd Ft.Meade,MD 20755-6704 410-854-6632 DSN: 244-6632 FAX: 410-854-6604 www.nsa.gov/snac
SNAC@radium.ncsc.mil

XML Denial of Service Issues - XML is a versatile data-encoding standard. However, parsing XML can be processor intensive and complex, which can lead to security issues. One common issue is a denial of service (DOS) against a web service. If an attacker crafts an XML message with very large payloads, recursive content, excessive nesting, malicious external entities, or with malicious DTDs (Data Type Documents), a DOS can occur.

Mitigations - When processing XML, use filters, XML Gateways, or XML parser options to prevent parsers from processing malicious messages. Filters, gateways, or parser options can restrict the rate of messages per second, the message size, the number of nested XML elements, the number of XML attributes, the number of XML nodes, the length of an XML node name, DTDs, and external entities. XML is hard to parse correctly so only use proven, robust XML parsers. Developers and system administrators sometimes use schema validation as a way to validate XML input. Schema validation is of limited help with malicious XML because the parser must parse the XML before validating the document.

Insecure Communications – Attackers can steal or modify information if not protected while in transit.

Mitigations – Use the latest versions of SSL or TLS, version 3 and 4 respectively as of this writing, to protect the content of messages in point-to-point transactions. Requiring mutual authentication between the client and server raises the level of trust before processing messages and generally decreases the attack surface of the service. Web Services allow for messages to be routed though multiple intermediaries; one of the intermediaries may terminate the SSL or TLS connection so the message may not be protected between all of the intermediaries. In these architectures, use end-to-end security mechanisms like XML-Encryption, XML-Signature, and SAML assertions (Security Assertion Markup Language). End-to-end XML security mechanisms are complicated and the implementation must be reviewed and tested to ensure the protection is adequate.

Information Leakage - Web Services that generate verbose fault messages are useful to

developers and system administrators. However, the same messages can give away too much information in operational environments. This issue also affects Web Services that use a WSDL to provide a description of a service and its interface. A WSDL contains server directory information, internal IP address information, available services and methods, and other critical information valuable to an attacker.
Mitigations - System administrators should configure servers to minimize the leakage of information by not advertising software specifics, removing WSDL's or authenticating the user before sending the WSDL, and turning off all verbose error messages.

Replay Attack Flaws – Protecting a message against modification does not stop an attacker from replaying the message to a server to invoke actions multiple times.
Mitigations - Encryption and digital signatures may provide protection against eavesdropping and modification; however, if an encrypted or signed message can be intercepted, it may still be vulnerable to a replay attack. Developers can mitigate replay attacks in two ways: signed timestamps and nonces. Clients can include signed timestamps in the messages and servers can implement time windows. In addition to signed timestamps, clients can include signed unique message identifiers (nonces) so servers can implement message tracking (sometimes called a nonce pool) to keep track of the unique messages that have already been processed.

Insufficient Authentication - Web Services that perform sensitive functions should require authentication.
Mitigations - For any sensitive transaction, each request should be associated with an authenticated identity and each service or data provided should be associated with authorization rules. Passwords are typically not appropriate with these technologies. Instead, use PKI, multi-factor authentication, or the newer XML security based technologies like XML-Signature and SAML.

These vulnerabilities are common in other technologies, but become more critical with Web Services because of their interoperability and ability to circumvent firewalls.

Inadequate Testing – Unidentified coding flaws in Web Services can lead to a compromise of sensitive information. Because SOA implementations typically connect

to backend servers, the consequences of a compromise are amplified.
Mitigations – Test all Web Service code for vulnerabilities; a developer's mistake could lead an attacker to sensitive information. Testing should include code reviews, internal and third party penetration testing, and a quality assurance process. Project managers should include testing as part of the software development process; not as an afterthought.

Insecure Configuration – Web Services typically run on exposed, public facing servers, outside an organization's security perimeter. Mistakes in configurations and patch management of these servers can be catastrophic.
Mitigations - Whenever possible, separate Web Applications from Web Services by hosting each on different servers or ports. Because Web Services can be accessible to unauthenticated users, systems administrators must understand all configuration options and be diligent with configuration management and patch management.

Insufficient Logging - Logs are of great use if an intrusion or hacking attempt occurs.
Mitigations – System administrators and developers should log events such as unsuccessful and successful authentication attempts, unsuccessful authorization attempts, and application errors. It is also advisable to sign and encrypt the log files to protect their integrity and confidentiality. Logging without having a human look at the logs is useless; someone must review the log files at a regular interval or at least monitor them using automated log scanning or audit reduction tools.

SOA and Web Service technologies offer many benefits and are revolutionizing the way software interacts. However, with these benefits, comes additional risk. Organizations should be aware of the risks to Web Services so they can make informed decisions before deploying Web Services in a SOA.

References:

- OWASP Top 10 2007 – http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
- Shah, Shreeraj. Hacking Web Services. 2007.
- Kanneganti, Ramarao; Chodavarapu, Prasad. SOA Security. 2008.