# Modeling the Evaluation Criteria for Security Patterns in Web Service Discovery

V.Prasath
K.C.E.T, Department of IT
S.Kumarapuram, Cuddalore
India 607109.

## ABSTRACT
Current trends in performing business-to-business transactions and enterprise application integration have been extended to the use of web service. With web services being accepted and deployed in both research and industrial areas, the security related issues become important. Web services security has attracted the attention of researchers in the area of security due to the proven fact that most attacks to businesses and organizations exploit web service vulnerabilities. The main goal of this research is to achieve security concept of the web service can be summarized to this single value. In this paper, we evaluate common security patterns with respect to the STRIDE model of attacks by examining the attacks performed in different web services system. In order to evaluate security we introduce a new measure for the computation of a security rating of web service based on STRIDE test case model such that the security concept of the system can be summarized to single value. The overall severity for the risk that can expressed in measurable way.

## Categories and Subject Descriptors
D.3.3 [**Information Systems Applications]:** *Metrics-security measures*

## General Terms
Security

## Keywords
Risk analysis • Stride • WSSecurity • Security rating •Web service discovery

## 1. INTRODUCTION
Web services are one of the most promising technologies for building distributed systems that has the potential of becoming the core of a new Web-based middleware platform, providing interoperability between computational services. In this specific context security is very important feature. Nowadays, many companies and organizations implement their core business and application services over Internet. Thus, the ability to efficiently and effectively select and integrate inter-organizational and heterogeneous services on the Web at runtime is an important step towards the development of the web service applications [1].

Web services communication is based upon the Simple Object Access Protocol (SOAP). SOAP is an XML-based information packaging definition which can be used for exchanging structured and typed information between peers in a distributed environment, relying on Internet transport protocols such as HTTP. Because SOAP is standards based, it also provides interoperability in heterogeneous environments. A large number of web services are being developed as an emerging standard to construct distributed applications in the web. Service requesters have access to a choice of descriptions to various services that provide similar service functionality. Automation of dynamic web service discovery is made viable by expression of domain specific knowledge [2] [4]. Today's systems, and the enterprises in which they reside, are so complex that even the most capable risk measurement tools are unlikely to yield risk values that are much better than rough indications of relative risk which, we should quickly add, is often quite good enough in many situations. The problem is that the value of risk, whatever it turns out to be, is likely to be surrounded by a fairly large but unknown amount of uncertainty. This can create a dilemma for the decision-maker who must then decide whether to invest in further safeguards, which will undoubtedly reduce the overall risk but could be both expensive and unnecessary, or to collect more evidence to reduce the amount of uncertainty surrounding the risk calculation [3].The high importance of web services security to the process of ensuring some level of security to real systems has been evident since it has been discovered that most attacks may exploit web vulnerabilities [5, 6, 7]. These vulnerabilities stem from web service are poorly designed and developed. Therefore, the incorporation of a level of security already at the design phase is desirable.

If multiple Web services provide the same functionality, then a security requirement can be used as a secondary criterion for service selection. Security is a set of non-functional attributes like confidentiality, integrity, availability. The current Universal Description, Discovery and Integration (UDDI) registries only support Web services discovery based on the functional aspects of services [8]. The problem, therefore, is firstly to accommodate the security information in the UDDI, and secondly to guarantee some extent of authenticity of the published information. Security information published by the service providers may not always be accurate and up-to-date. To validate security promises made by providers, we propose a new system to rate the various security attributes of the Web services they use. These ratings are then published to provide new customers with valuable information that can be used for services selection. We concentrate here on one key issue, providing security for web services in dynamic nature. To realize the potential risks which can arise if proper counter measures are not implemented, it is important to be able to determine the security capabilities of web services in order to ensure that the system is protected against such risks. To do this, some type of security evaluation framework is needed, such that this framework can be used to summarize all implemented concepts to a handy rating [9].

We try to practically examine the resistance to STRIDE attacks of a small subset of security patterns that are commonly used in web service applications. To perform this evaluation, we have to built a system with web services security testing patterns and using them to study systems under known categories of attacks to web service applications [10] and determine which aspects of security are enhanced through the use of each security

pattern for web service system. To study these systems under known attacks, we have used different testing methodology approaches based on the SOAP Sonar for web service penetration testing tool [11] that aim to evaluate web service security in terms of security vulnerabilities. Finally, based on our findings we determined to what extent each security pattern protects us from each category of STRIDE [7] attacks. Based on the fact that we can not quantify in strict terms the security of a system [6] levels of security ranging from absolutely low to absolutely high have been used in the analysis instead of exact numbers. The experimental evaluation shows that each security pattern protects us from different categories of attacks and therefore a smart combination of these security patterns, based on the resistance of each pattern to each category of attack, can lead to systems that are secure enough already from their design. This paper will describe a testing methodology for web services security and outline a process that can be adopted to evaluate web services security can be summarized to single value. We define security as a measure of vulnerabilities in the accuracy of a risk or security measurement. In order to render the definition useable, we believe it is necessary to associate the terms in the definition with a measurement scale that represents the security of a system as a value between 0 (secure) and 1 (insecure).

## 2. RELATED WORK

Web service providers must assure their clients confidentiality, integrity and availability over a trusted relationship that may be asynchronous and that may involve multiple business partners. There has been no work on a formalised security testing methodology specifically targeted at web services. However, a testing framework aimed at standard web applications exists and provides a very informative starting point.

Several authors conducted performance evaluation studies of WS-Security. In [12] and also [13] the authors compare the performance of WS-Security operations and choices of signature and encryption algorithms to non-secure messages using various message sizes and complexities. WS-Security provides end-to-end security properties (integrity, confidentiality, and authentication) through open XML standards. End-to-end message security assures the participation of non-secure transport intermediaries in message exchanges, which is a key advantage for Web-based systems and service-oriented architectures. However, point-to-point message security based on TLS (Transport Layer Security) is known to significantly outperform WS-Security.

The security characteristics of web service based systems depend on those of the individual web services (WS) involved and the way in which they are related to each other. In principle, the security characteristics of WS or systems can be expressed in security properties that are published and available to external parties. Since SOAP itself does not provide secure transmission protocol for messages, it brings high risks to both sides of the message exchanger. Although traditional security technologies such as SSL and HTTPS can partially resolve this problem by encrypting messages transferred between two points [1], these point-to-point transport-layer security technologies cannot insure end-to-end security along the entire path from client to a web service in a complicated multi-tiers distributed system. Furthermore, these point-to-point security technologies are all based on a specific transport protocol/layer, such as TCP/IP for SSL and HTTP for HTTPS. Since SOAP is a transport-independent messaging protocol for web services, the capacity and application of web services would be limited if its security relies on these transport dependent technologies. As a result, OASIS developed Web Services Security (WSS) specification [14] to provide message-level protection between two ends (clients and web services) through message integrity, message confidentiality and message authentications. WSS makes use of SOAP's composable and extendable architecture by embedding security-related information (security token, signatures etc.) in the SOAP header without affecting the data stored in the SOAP's body (but maybe encrypted/signed). This design allows WSS to integrate with SOAP as a plug-in and still retain SOAP's composability and extensibility for other purposes. Today more and more web services products are beginning to support the WSS standard [15][16]. While WSS enhances the security of web services, people may be concerned with its performance overheads. The overheads can come from: (a) extra CPU times to process WSS-related elements; (b) longer networking times to transport larger SOAP messages due to additional WSS contents [17][18][19]. Although many application security testing principles can be generically applied to web services, particular aspects of the technology such as its reliance upon XML and web services specific standards require closer attention that is not provided by other testing methodologies. Thus, a comprehensive framework for evaluating the security of web service implementations.

With web services being accepted and deployed in both research and industrial areas, the security related issues become important. The demand for web services and applications in cyberspace is hindered by security concerns that are raised by corporate service providers and service users. There are concerns about the trustworthiness of the web services from both sides of the spectrum. Testing web services security is a critical step towards enhancing their trustworthiness. To address these issues, we propose a comprehensive framework for specifying security requirements for web services. In this paper, we introduce a new measure for the computation of a security rating of web services based on ws-security test case model such that the security concept of the system can be summarized to single value. These security concepts are used in the measure computation to get the security rating of the web services based on STRIDE methodology.

## 3. WEB SERVICES TECHNOLOGIES

Web service architecture involves many layered and interrelated technologies. There are many ways to visualize these technologies, just as there are many ways to build and use Web services. Figure1 below provides one illustration of some of these technology families. In this section we describe some of those technologies that seem critical and the role they fill in relation to this architecture. This is a necessarily bottom-up perspective, since, in this section, we are looking at Web services from the perspective of tools which can be used to design, build and deploy Web services. The technologies that we consider here, in relation to the Architecture, are XML, SOAP, and WSDL. However, there are many other technologies that may be useful.

### 3.1 XML

XML solves a key technology requirement that appears in many places. By offering a standard, flexible and inherently extensible data format, XML significantly reduces the burden of deploying the many technologies needed to ensure the success of Web services. The important aspects of XML, for the purposes of this Architecture, are the core syntax itself, the concepts of the XML

Infoset [XML Infoset], XML Schema and XML Namespaces. XML Infoset is not a data format per se, but a formal set of information items and their associated properties that comprise an abstract description of an XML document. The XML Infoset specification provides for a consistent and rigorous set of definitions for use in other specifications that need to refer to the information in a well-formed XML document. Serialization of the XML Infoset definitions of information may be expressed using XML. However, this is not an inherent requirement of the architecture. The flexibility in choice of serialization format(s) allows for broader interoperability between agents in the system.

## 3.2 SOAP

SOAP 1.2 provides a standard, extensible, compostable framework for packaging and exchanging XML Messages. In the context of this architecture, SOAP 1.2 also provides a convenient mechanism for referencing capabilities. SOAP 1.2 defines an XML-based messaging framework: a processing model and an extensibility model. SOAP messages can be carried by a variety of network protocols; such as HTTP, SMTP, FTP, RMI/IIOP, or a proprietary messaging protocol. SOAP 1.2 defines three optional components: a set of encoding rules for expressing instances of application-defined data types, a convention for representing remote procedure calls (RPC) and responses, and a set of rules for using SOAP with HTTP/1.1. While SOAP Version 1.2 doesn't define "SOAP" as an acronym anymore, there are two expansions of the term that reflect these different ways in which the technology can be interpreted:

1)Service Oriented Architecture Protocol: In the general case, a SOAP message represents the information needed to invoke a service or reflect the results of a service invocation, and contains the information specified in the service interface definition.

2) Simple Object Access Protocol: When using the optional SOAP RPC Representation, a SOAP message represents a method invocation on a remote object, and the serialization of in the argument list of that method that must be moved from the local environment to the remote environment.


**Figure 1. Web Service Technologies**

## 3.3 UDDI

Universal Description, Discovery and Integration (UDDI) is a platform-independent, XML-based registry for businesses worldwide to list themselves on the Internet. UDDI is an open industry initiative, sponsored by OASIS, enabling businesses to publish service listings and discover each other and define how the services or software applications interact over the Internet. A UDDI business registration consists of three components:

White Pages — address, contact, and known identifiers

Yellow Pages — industrial categorizations based on standard taxonomies

Green Pages — technical information about services exposed by the business

UDDI was originally proposed as a core Web service standard. It is designed to be interrogated by SOAP messages and to provide access to Web Services Description Language documents describing the protocol bindings and message formats required to interact with the web services listed in its directory.

## 3.4 WSDL

WSDL 2.0 is a language for describing Web services. WSDL describes Web services starting with the messages that are exchanged between the requester and provider agents. The messages themselves are described abstractly and then bound to a concrete network protocol and message format. Web service definitions can be mapped to any implementation language, platform, object model, or messaging system. Simple extensions to existing Internet infrastructure can implement Web services for interaction via browsers or directly within an application. The application could be implemented using COM, JMS, CORBA, COBOL, or any number of proprietary integration solutions. As long as both the sender and receiver agree on the service description, the implementations behind the Web services can be anything.

## 3.5 SSL/HTTPS

Secure Sockets Layer (SSL) was developed by Netscape and is now used by all web servers and browsers as tool for authentication and encryption. It runs "on top of" TCP/IP but "underneath" transports such as HTTP and LDAP [12]. HTTP is run over SSL to create what is commonly called "HTTPS". Normally, HTTPS is used to validate the identity of the server to the client (via a chain of trusted certificates), and provides end-to-end encryption for the HTTP protocol. HTTPS can also provide client authentication, but HTTP basic authentication (explanation directly below) is normally used for that. HTTPS is often used to provide confidentiality (via strong encryption) for web services, but it's not a complete solution to the problem.

## 3.6 HTTP BASIC AUTHENTICATION

Basic authentication is often used to provide userid/password authorization to web resources. All web servers provide a means for protecting resources using basic authentication. Using this technique, the web server checks to see if the user has sent authentication credentials when trying to access a protected resource. If the user has not logged in (indicated by the presence of base-64 encoded credentials in the HTTP header), he is challenged with a login dialog. Basic authentication is not considered strong authentication, because it's trivial to unencode the username and password (Base64 encoding is not considered encryption, because there's no secret key). HTTP digest authentication is also available, and is more secure than basic authentication, but it is not widely used, and support in browsers is inconsistent.

## 3.7 SAML

It may not be obvious at first, but one key aspect of web services is the concept of single sign-on. The idea is that once the user has authenticated once, an application (or another web service) can forward that user's credentials to another web service and request action on the user's behalf. For this to work, we need a method for encoding the "assertion" by one application that it has authenticated the user's credentials, allowing it to pass those credentials securely. An assertion is a declaration of facts (statements) about a subject (according to some SAML authority).

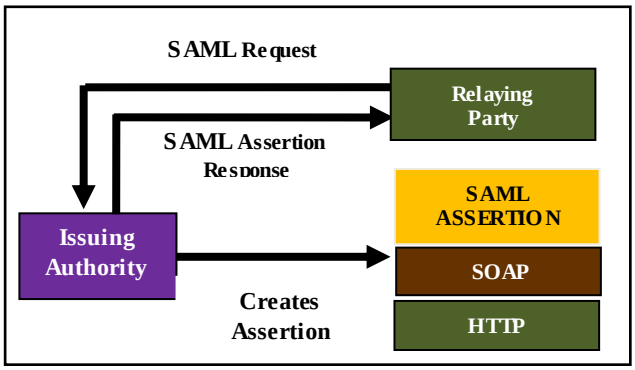


**Figure 2. XML for Access Control Lists (XACL)**

Assertions can also be digitally signed, to prove that they came from a trusted source. In addition, we need the ability to describe what parts of an application the user is authorized to access. SAML enables applications (web services) to exchange identity and entitlement information with each other. SAML works by exchanging "assertions", which confirm information about a user's authorization, authentication, or other information. These assertions are usually time-bound, and describe events that have already occurred, such as, "This user has been successfully authenticated", or "This application is authorized to take this action." Here's an example SAML authentication assertion, taken directly from.

## 3.8 XACL

XACL allows you to hide or expose certain portions of an XML document from users, based on their roles. The idea is simple, in addition to the XML document you want to protect, define an extra document that references the original, and constrains access by role.

# 4. EVALUATION OF RISK CERTAINTY

There are many different approaches to risk analysis. Our approach presented here is based on these standard methodologies and is customized for web services security testing methodology based on STRIDE attacks [22]. In this paper a new measure for the security of web services will be presented. For example [20], [Weippl, 2005] proposes to list the important assets and the possible risks in a first step. In the second step, integers between 1 and 10 have to be assigned to these risks and assets. For every pair of asset and risk, one integer represents the probability that a security risk gets real and a second integer describes the impact of such a security threat to the asset. For combining the integers to get the value of protecting the assets against such a security threat. In our approach, a catalogue for relevant criteria for testing to be considered will be presented, and second, the computation of a security rating by considering the concept tested against the security criteria will be given based on STRIDE attacks. For introducing a rating corresponding to the security level of some criteria, we give a first try to evaluate the criteria. There are a number of factors that can help us to find raring. The set of factors involved in identifying risk based on threat, vulnerabilities and technical impact. Each factor has a set of options, and each option has a likelihood rating from 0 to 3 associated with it. We will use these numbers later to estimate the overall likelihood. The first step is to identify a security risk that needs to be rated. Once identified a potential risk, and want to figure out how serious it is, in order to estimate the "likelihood". At the highest level, this is a rough measure of how likely this particular vulnerability is to be uncovered and exploited by an attacker. Generally, identifying whether the likelihood is low (1), medium (2), or high (3) is sufficient.

# 5. EVALUATION OF SYSTEM UNDER STRIDE PATTERN

Aim of this paper is to introduce a new measure of computation for the security of web services. For this purpose the implemented criteria as mentioned below have to be considered and integrated into the calculation according to their security strength. Before we can state some formula for the calculation process, we have to introduce the factors needed for it. STRIDE is an acronym for a process developed by the Microsoft Application Consulting and Engineering Team to represent various methods by which an adversary may attack a system. Threats are often classified according to their type. Threat types are often associated with specific security mechanisms and can therefore be quickly mitigated. One form of classification is known as the STRIDE model. To create a catalogue of security criteria, we consider a model for security services, as given by a combination of the services presented in [Voydock/Kent, 1983] and [ISO, 1989] [21]. These security services that must be enforced in order to create a secure environment are listed in the following:

Spoofing: In this attack, adversaries falsely represent themselves as valid user entities. For example, having obtained the login of a system administrator, the attacker gains access to system data, giving them free rein to execute further attacks.

Tampering: Using this method of attack, an adversary successfully modifies or deletes data within the system. An example would be when an adversary gains access to the system database and deletes all the client records.

Repudiation: This method identifies whether or not an adversary can attack a system without detection or evidence that the attack occurred. An example would be an adversary who performs a "tampering with data" attack without leaving any trail indicating that the data had been compromised.

Information Disclosure: In this attack method, an adversary gains access to data not within their trust level. Such data may include system information that may facilitate further attacks.

Denial of Service: Using this method of attack, an adversary causes a system to be unavailable for valid user entities. An example would be an adversary who executes a shutdown command to a file server.

Elevation of Privilege: This type of attack increases the adversary's system trust level, permitting additional attacks. An example would be an adversary who enters a system as an anonymous user entity but is able to obtain the trust level of a system administrator.

In order to evaluate web services security under known attacks we have used SOAP Sonar [11] web services penetration testing tool. Bringing this approaches to find attacks that found the major security flaws of the web service application, meaning the three Sub-system Parameters, two SQL Injection, three authentication exchange tampering and three replay attacks Minor application errors that pose no threat to security not found by the static approach, were found by SOAP Sonar [11].

Additionally, the threat attacks found in the remaining web services were fewer in comparison to the second application and first one. The higher number of security flaws in the first application was much more prominent in the set of high risk flaws. The previous analysis of the results shows that proper use of the security patterns leads to the remediation of all major security flaws. These flaws that remain even after the use of security patterns exist because existing security patterns do not confront these kind of problems. The intercepting validator pattern, when used for all input, including session variables that are not input by the user but still posted, protects from Sub-system Parameters ,SQL Injection, authentication exchange tampering and replay attacks. Therefore, it offers high protection against Tampering with Data and Information Disclosure attacks.

The STRIDE model has been adopted for the purpose of threat modelling in this paper. The following is a generic STRIDE threat classification for web services which should be tailored for the domain of the specific web service being tested. Each threat is identified with a reference number and contains a description, possible architecture entry points and a list of assets that may be impacted. STRIDE is a very simple approach to threat identification. The terms/phrases it represents, along with an explanation of each, are listed in [22] Table 1. At each trust boundary (TB), apply the STRIDE model by asking whether one or more of the threat types represented to apply. Because of its simplicity, its use tends to result in one or missed threats per TB.

**Table 1. Stride Pattern**

| STRIDE (MSDN) | |
|---|---|
| **Letter** | **Stands for…** |
| S | **Spoofing Identity** - Impersonating someone else to the computer |
| T | **Tampering with Data** - The malicious modification of data |
| R | **Repudiation** - Involves users who can deny performing an action without other parties having any way to prove otherwise |
| I | **Information Disclosure** - Involves the exposure of informaion to individuals who are not supposed to have access to it |
| D | **Denial of Service** |
| E | **Elevation of Privilege** - An unprivileged user gains privileged access and thereby has enough access to compromise or destroy the system |

## 6. CATEGORIZE AND PRIORITIZE THREATS

In an organization where threat and vulnerability management is governed by solid risk management principles, the following formula is typically used to assign a risk score to a threat:

**Risk = Probability of Occurrence x Business Impact**

There are a number of ways, both qualitative and quantitative, to apply this formula. For the purposes of our threat assessment model, I'm going to use STRIDE. STRIDE is a classification scheme for characterizing known threats according to the kinds of exploit that are used (or motivation of the attacker). At this stage we have a list of threats that apply to the application. In the final step of the process, you rate threats based on the risks they pose. This allows you to address the threats that present the most risk first, and then resolve the other threats. In fact, it may not be economically viable to address all of the identified threats, and you may decide to ignore some because of the chance of them occurring is small and the damage that would result if they did is minimal.

$$\textbf{Risk} = \frac{\textbf{No. of Inbound Attacks + No.of OutBound Attacks}}{\textbf{Total no. of Assests}}$$

This formula indicates that the risk posed by a particular threat is equal to the probability of the threat occurring multiplied by the damage potential, which indicates the consequences to your system if an attack were to occur. You can use a 1–3 scale for probability where 1 represents a threat that is very unlikely to occur and 3 represents a near certainty. Using this approach, the risk posed by a threat with a low likelihood of occurring but with high damage potential is equal to the risk posed by a threat.For example, if **InBound** =3 and **OutBound** =1 for replay attacks, then Risk = 3 * 1 = 3. If **Overall**=3 then risk rating that occur in the value in the scale be high. This approach results in a scale of 1–3, and you can divide the scale into three bands to generate a High, Medium, or Low risk rating.

**Table 2. Test Case Threat Mapping**

| Sl. No | Parameter | Asset | Threat | Description | Measurement |
|---|---|---|---|---|---|
| 1 | Wsdl Scanning | Information gathering | Information Disclosure | It describing the functionality offered by the web service and the parameters required to use it. | Rating |
| 2 | Web Method Enumeration | Information gathering | Information Disclosure | Not all implemented methods may be published in the WSDL document | Rating |
| 3 | Error message Information Leakage | Information gathering | Information Disclosure | Error messages within SOAP faults can contain detailed platform information and implementation details such as code fragments or stack traces. | Rating |
| 4 | Numerical Values | Fuzzing | Information Disclosure | Any value that is only as a numerical value or is expected to be a numerical value. | Rating |
| 5 | Base64 Encoded Values | Fuzzing | Tampering | Base64 is used to encode binary data in order to conform to XML specifications. | Rating |
| 6 | Character Strings | Fuzzing | Tampering | This verybroad category general guidelines for any data that is not of any particularly classifiable form. | Rating |
| 7 | General values | Fuzzing | Tampering | If it is not possible to indentify the nature of the values beings supplied this category provides a general overview of the types of inputs that should be tested. | Rating |
| 8 | Sub system parameter | Fuzzing | Spoofing | This category relates to any values that may used to influence output on the | Rating |

| | | | | | |
|---|---|---|---|---|---|
| | | | | client side of the application. | |
| 9 | Addressing parameters | Fuzzing | Tampering | System often use addressing information to access information directories. | Rating |
| 10 | Logging values | Fuzzing | Tampering | Any value that is logged directly to some medium has the potential to somehow corrupt logs or provide an inaccurate view. | Rating |
| 11 | Sql Injection | Injection | Spoofing | Any value that may be used as part of an SQL query should be tested for the ability to change SQL processing in some way,possibly causing data disclosure. | Rating |
| 12 | Command Injection | Injection | Tampering | If an internal system is used to execute existing commands and input to these commands is not properly validated,it may be possible to run commands of the user's choosing. | Rating |
| 13 | Lpath Injection | Injection | Spoofing | If LDAP queries are constructed directly from user input, this may result in significant system compromise,particularly in the disclosure of user credentials. | Rating |
| 14 | Xpath Injection | Injection | Information Disclosure | The use of user supplied input in an XPath query may provide an attacker with the ability to modify the query. | Rating |
| 15 | Code Injection | Injection | Elevation of Privileges | If an validated user supplied input is supplied to calls to eval-type functions, malicious commands may be inadvertently executed by the web service | Rating |
| 16 | Cipher choice | Confidentia lity | Information Disclosure | The choice of encryption cipher will influence the strength of the encryption and the ability for an attacker to successfully crack the encryption and recover plaintext data | Rating |
| 17 | Encryption Coverage | Confidentia lity | Information Disclosure | Encryption should be applied overall sensitive portions of messages to ensure they are protected against un authorized eaves dropping.. | Rating |
| 18 | Replay Attacks | Integrity | Spoofing | A replay attack involves the malicious use of a valid message or set of messages that has already been accepted by the web service previously. | Rating |
| 19 | Integrity Check Coverage | Integrity | Tampering | Integrity checks should be used to protect important data against unauthorized modification. | Rating |
| 20 | Invalid Xml | Integrity | Denial of service | WS-Security and other web sevice security standards are XML-based and their implementations require properly formed XML to function properly. | Rating |
| 21 | Unsupported algorithms | Integrity | Tampering | Verify that if unsupported algorithms are requested or the client claims to root support required algorithms,access is denied and processing of the request does not continue. | Rating |
| 22 | Separator Injection | Logging | Repudiation | Log entries are commonly delimitedusing a particular separator character. | Rating |
| 23 | White Space Injection | Logging | Repudiation | White space characters can be used to modify the appearance of log entries when they are viewed. | Rating |
| 24 | Brute-Force and Dictionary Attacks | Authenticati on | Elevation of Privileges | These types of attacks are typically used against password authentication systems and rely on the ability to repeatedly test potential passwords against the authentication service. | Rating |
| 25 | Forged Credentials | Authenticati on | Elevation of Privileges | Credentials should be issued by an authorized party and verified by the application when presented. | Rating |
| 26 | Missing Credentials | Authenticati on | Spoofing | A user that fails to present credentials should not be allowed access and the application should discard their request. | Rating |
| 27 | Token Forgery | Authorizati on | Elevation of privileges | As SOAP is a stateless message-based protocol ,some mechanism must be implemented to provide authorization between SOAP requests or maintain session state. | Rating |
| 28 | Hijacking Attacks | Authorizati on | Tampering | As SOAP is a stateless message-based protocol, some mechanism must be implemented to provide authorization between SOAP requests or maintain session state. | Rating |
| 29 | Parameter Tampering | Availability | Denial of service | This broad class of attacks refers to the modification of SOAP request parameters in transit between client and server. | Rating |
| 30 | Coercive Parsing | Availability | Denial of service | Coercive parsing is the name given to the class of attacks that involve supplying illegal or malformed SOAP requests to the web service in order to cause undesirable behavior. | Rating |

**High, Medium and Low Rating**

You can use a simple High, Medium, or Low scale to prioritize threats. If a threat is rated as high, it poses a significant risk to your application and needs to be addressed as soon as possible. Medium threats need to be addressed, but with less urgency. You may decide to ignore low threats depending upon how much effort and cost is required to address the threat.

In– No. of Agent attacks
Out – No. of Masquerader attacks
1- Low 2-Medium 3-High

To evaluate web services security under known attacks we have used SOAP Sonar web services penetration testing tool [11]. Bringing this approaches to find attacks that found the major security flaws of the web service application threat possible to occur. Based on the above analysis that offers us a practical examination of attacks to web service with security patterns. We can make conclusions about the resistance to known categories of attacks of the security patterns under consideration.

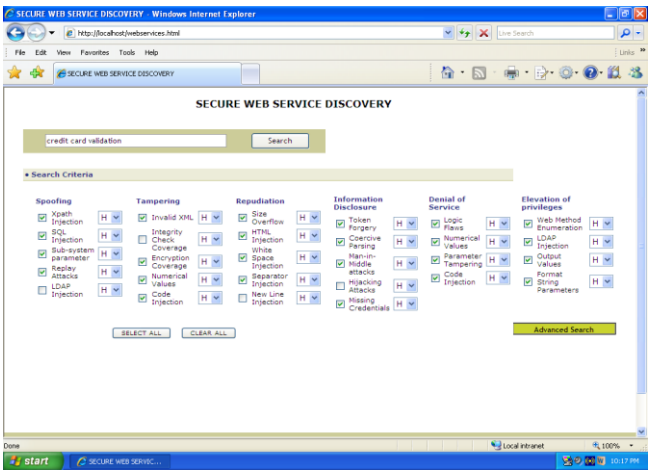


**Figure 3. Criteria for Secure Web Service Discovery**

**Table 3. Spoofing Attacks**

| Elevation of privileges(E) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Web Service | No. of Source Attacks | | | | | | | | Overall |
| | Web Method Enumeration | | Output Values | | LDAP Injection | | Format String Parameters | | |
| | In | Out | In | Out | In | Out | In | Out | Total |
| Dictionary | 1 | 3 | 2 | 2 | 2 | 1 | 2 | 2 | 15 |
| GetWeather | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 21 |
| My Service | 1 | 3 | 2 | 2 | 2 | 2 | 1 | 1 | 14 |
| GetJoke | 1 | 1 | 3 | 3 | 0 | 2 | 2 | 3 | 15 |

**Table 4. Tampering Attacks**

| Web Service | S | T | R | I | D | E | Overall | Rank |
|---|---|---|---|---|---|---|---|---|
| Dictionary | 16 | 11 | 10 | 13 | 10 | 15 | 0.46 | 1 |
| GetWeather | 13 | 16 | 12 | 17 | 8 | 21 | 0.54 | 4 |
| My Service | 14 | 12 | 17 | 12 | 12 | 14 | 0.5 | 2 |
| GetJoke | 12 | 12 | 14 | 11 | 19 | 15 | 0.51 | 3 |

**Table 5. Repudiation Attacks**

| Repudiation(R) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Web Service | No. of Source Attacks | | | | | | | | Overall |
| | White space injection | | Separator Injection | | HTML Injection | | Size Overflow | | |
| | In | Out | In | Out | In | Out | In | Out | Total |
| Dictionary | 1 | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 10 |
| GetWeather | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 3 | 12 |
| My Service | 2 | 1 | 3 | 2 | 3 | 3 | 1 | 2 | 17 |
| GetJoke | 2 | 2 | 2 | 1 | 2 | 1 | 3 | 1 | 14 |

**Table 6. Information Disclosure Attacks**

| Spoofing(S) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Web Service | No. of Source Attacks | | | | | | | | Overall |
| | Sub-system Parameters | | SQL Injection | | Xpath Injection | | Replay Attacks | | |
| | In | Out | In | Out | In | Out | In | Out | Total |
| Dictionary | 1 | 3 | 2 | 2 | 1 | 1 | 3 | 3 | 16 |
| GetWeather | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 13 |
| My Service | 1 | 1 | 3 | 3 | 2 | 2 | 1 | 1 | 14 |
| GetJoke | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 1 | 12 |

**Table 7. DoS Attacks**

| Tampering(T) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Web Service | No. of Source Attacks | | | | | | | | Overall |
| | Numerical Values | | Code Injection | | Invalid XML | | Encryption Coverage | | |
| | In | Out | In | Out | In | Out | In | Out | Total |
| Dictionary | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 1 | 11 |
| GetWeather | 1 | 1 | 3 | 1 | 3 | 2 | 2 | 3 | 16 |
| My Service | 1 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 12 |
| GetJoke | 2 | 2 | 1 | 1 | 2 | 1 | 2 | 1 | 12 |

**Table 8. Elevation of privileges Attacks**

| Information Disclosure(I) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Web Service | No. of Source Attacks | | | | | | | | Overall |
| | Token Forgery | | Coercive Parsing | | Man in Middle attack | | Missing Credentials | | |
| | In | Out | In | Out | In | Out | In | Out | Total |
| Dictionary | 1 | 2 | 1 | 2 | 1 | 1 | 3 | 2 | 13 |
| GetWeather | 3 | 1 | 2 | 1 | 2 | 2 | 3 | 3 | 17 |
| My Service | 1 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 12 |
| GetJoke | 1 | 2 | 0 | 1 | 3 | 1 | 3 | 0 | 11 |

**Table 9. Overall Security**

| Denial of Service(D) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Web Service | No. of Source Attacks | | | | | | | | Overall |
| | Numerical Values | | Code Injection | | Logic Flaws | | Parameter Tampering | | |
| | In | Out | In | Out | In | Out | In | Out | Total |
| Dictionary | 1 | 2 | 1 | 2 | 0 | 1 | 2 | 1 | 10 |
| GetWeather | 1 | 1 | 1 | 1 | 2 | 0 | 2 | 0 | 8 |
| My Service | 1 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 12 |
| GetJoke | 2 | 2 | 1 | 2 | 3 | 3 | 3 | 3 | 19 |

**Calculation of Risk Value based on STRIDE**

$$\textbf{Risk} = \frac{\textbf{No. of Inbound Attacks + No.of OutBound Attacks}}{\textbf{Total no. of Assests}}$$

$$\textbf{Risk (}\text{Spoofing)} = \frac{\{\text{Inbound}((1+2+1+3)+(1+1+1+2)+(1+1+0+2)+(1+2+2+2))\} + \{\text{Outbound}((3+2+1+3)+(2+2+1+1)+(2+2+1+1)+(3+2+1+1))\}}{16}$$

$$= 0.46$$

## 7. CONCLUSIONS AND FUTURE DEVELOPMENTS

The problem with a simplistic rating system based on stride and dread model usually will not agree on rating value [23]. To overcome this challenges issues we proposed new type of rating scenario based on attacks that existing under different system based on criteria based security patterns. In order to evaluate the web services security under known attacks we have used STRIDE approach. We have estimated the resistance of specific security patterns in web services system against STRIDE attacks in measurable way. In order to achieve this we have built a system wit respect to security that generally applied to web service. Thus the increasing use in the enterprise sector for the integration of distributed systems and business critical functions dictates the need for security assurance yet there is currently no security testing methodology specifically adapted to applications that implement web services. This paper will also describe a testing methodology for web services security and outline a process that can be adopted to evaluate web services security. Future work includes proposing new security patterns for the security flaws that our analysis showed that existing security patterns do not confront, build using a mathematical model for the security of systems under STRIDE.

## 8. REFERENCES

[1] Bin Xu, Tao Li, Zhifeng Gu, Gang Wu "Quick Web Service Discovery and Composition in SEWSIP", Proceedings of the 8th IEEE International Conference on E-Commerce Technology and the 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06).

[2] Garofalakis, J., Panagis, Y., Sakkopoulos, E., Tsakalidis, A., "Web Service Discovery Mechanisms: Looking for a Needle in a Haystack?", International Workshop on Web Engineering, 2004.

[3] Jeffrey R. Williams and George F. Jelen, "A Practical Approach to Measuring Assurance", Document Number ATR 97043, Arca Systems, Inc., 23 April 1998.

[4] Aabhas V. Paliwal, Nabil R. Adam, Hui Xiong, Christof Bornhövd "Web Service Discovery via Semantic Association Ranking and Hyperclique Pattern Discovery", Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence(WI 2006 Main Conference Proceedings) (WI'06).

[5] J. Viega and G. McGraw, Building Secure Software, How to Avoid Security Problems the Right Way, Addison Wesley, 2002

[6] G. Hoglund and G. McGraw, Exploiting Software, How to Break Code, Addison Wesley, 2004.

[7] M. Howard and D. LeBlanc, Writing Secure Code, Microsoft Press, 2002.

[8] Vu, L., Hauswirth, M., and Aberer, K. (2005). "QoSbased service selection and ranking with trust and reputation management". In Proc. of the Intl. conf. on Cooperative Information Systems (CoopIS), Agia apa, Cyprus.

[9] Colin Atkinson, Philipp Bostan, Oliver Hummel and Dietmar Stoll, "A Practical Approach to Web Service Discovery and Retrieval",IEEE International conference on Web Services (ICWS 2007).

[10] J.Scambray and M.Shema, Hacking Exposed Web Applications, McGrawHill, 2002

[11] http://www.softpedia.com/get/Authoring-tools/Authoring-Related/SOAPSonar-Enterprise-Edition.shtml

[12] S. Chen, J. Zic, K. Tang, and D. Levy. Performance evaluationand modeling of Web services security. In Proceedings of the IEEE International Conference on Web Services (ICWS'07), pages 431–438, 2007.

[13] H. Liu, S. Pallickara, and G. Fox. Performance of Web services security. In 13th Annual Mardi Gras Conference, Baton Rouge, Lousiana, USA, Feburay 2005.

[14] M. B. Juric, I. Rozman, B. Brumen, M. Colnaric, and M. Hericko. Comparison of performance of Web services,WS-Security, RMI, and RMI-SSL. Journal of Systems and Software, 79(5):689–700, 2006.

[15] S. Makino, K. Tamura, T. Imamura, and Y. Nakamura. Implementation and performance of WS-Security. Int. J. Web Service Res., 1(1):58–72, 2004.

[16] A. Moralis, V. Pouli, M. Grammatikou, S. Papavassiliou,

V. Maglaris. Performance comparison of Web services security: Kerberos token profile against X.509 token profile. In ICNS '07: Proceedings of the Third International Conference on Networking and Services, page 28, Washington,DC, USA, 2007. IEEE Computer Society.

[17] H. Liu, S. Pallickara, and G. Fox. Performance of Web services security. In 13th Annual Mardi Gras Conference, Baton Rouge, Lousiana, USA, Feburay 2005.

[18] Network Working Group. The Transport LayerProtocol Version1.1 (RFC4346). Available at http://www.faqs.org/rfcs/rfc4346.html.

[19] Oasis Consortium. WS-Security specification, 2004. Available from www.oasis-open.org.