International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 3– No.2, July 2012 – www.ijais.org

Building Trust for Web Services Security Patterns

V.Prasath
Assistant Professor
Dept. of Computer Science
PKIET,Karaikal,Puducherry

ABSTRACT
Security is a very important aspect for web service technology. Many people find the idea of creating security metrics to be a daunting task. Secure web service discovery aims at finding the best component services that satisfy the end-to-end security requirements between service consumer and service provider. The term "metrics or criteria" refers to specific objectives that have defined security measurement. It helps to select the most suitable security configuration according to a consumer business process and different levels of trust. In this paper, we presents a new pattern methodology for web service to address the security issues and propose a scalable security computation based on a heuristic approach which decomposes the complex problem into smaller sub problems that can be solved more efficient than the original problem. We define quality of service (QoS) in security as a set of security requirements a service provider guarantees. We identified several security parameters classified under different criteria to evaluate web services vulnerabilities. Metrics evaluation can be done through heuristic approach where in the security parameters are assigned prioritized weights which meliorates rank of web services.

General Terms
Web Service Security, Rating Evaluation

Keywords
Security patterns, Secure Service discovery, Trust, Security, Ranking

1. INTRODUCTION
Web services have acquired enormous popularity among software developers. This popularity has motivated developers to publish a large number of web service descriptions in UDDI registries. Although these registries provide search facilities, they are still rather difficult to use and often require service consumers to spend too much time manually browsing and selecting service descriptions. The availability of Internet and its services means that the information, the computing systems, and the security controls are all accessible and operable in committed state at some random point of time [1]. The inherent vulnerabilities of the internet architecture provide opportunities for a lot of attacks on its infrastructure and services. The growing emphasis on techniques for discovering relevant web services has dramatically increased the need for methods that let clients effectively find web services that are tailored to their requirements [2]. Such requirements can be functional (what the service offers) or non-functional (constraints on various properties such as quality of service, service reputation, interface semantics, and security). Because many web services will likely deliver similar functionalities, determining which are the most suitable without assessing their behavior under certain conditions will be challenging. Measuring the degrees to which web services can deliver the desired functionality through a combination of QoP (Quality of Protection) parameters becomes significant, particularly in distinguishing among services competing in the same domain. QoP encompasses a combination of security parameters that can help characterize web services overall behavior including authentication, confidentiality, integrity and availability. To illustrate the importance of incorporating QoP into the service discovery process, consider the following scenario.

A company is looking for a Web service to obtain localized weather information for its client management system. The company identifies certain characteristics that Web service must possess, including fast response time (under 50 ms), high availability (97 percent), and low transaction price (US$0.05 or less). The company performs manual search using existing UDDI registries and search engines and finds that at least 40 Web services offer the same functionality. After collecting price information from the corresponding service providers, the company determines that at least 20 services meet their requirements. Which service should the company choose? How much time and effort does it have to invest in finding relevant Web services? What distinguishes these Web services from each other? The choice could be simple if you assume that the service providers Quality of Service claims are trustworthy. In this case, the company will choose the Web services with fastest response time, highest availability and lowest transaction price [3]. If client focus on Quality of service in security, service selection becoming more and more important to choose a service that meet consumer needs. But so many vulnerabilities have been threatening web services, so it is meaningful to quantitatively evaluate them, which can reflects web services security reasonably and directly, and it will be convenient for user to choose service and deploy security measures.

The current UDDI standard and search engines can't answer these statements for many reasons, because service registries and search engines provide no Quality of Protection representation or support for Web services; so searching through service registries and search engines is insufficient. Although you could use web based search techniques to locate a web service, doing so involves some technical challenges. For example, Web pages often contain long textual information, whereas Web services have brief textual descriptions of what they offer or how to invoke them. This lack of detailed textual information increases the likelihood of keyword-based searches returning irrelevant search results. In addition, because Web pages primarily contain plain text, search engines can exploit information-retrieval methods, such as finding document and term frequencies. However, the structure of web services is much more complex than that of Web pages; Web services typically provide only a small portion of plain text, making basic information-retrieval techniques unreliable for Web service discovery. Furthermore,

interface information such as message, operation, and parameter names can vary significantly, so finding trends, relationships, or patterns within them is difficult and requires excessive domain knowledge in XML schemas and namespaces. The nature of web service discovery imposes additional requirements to the most common information-retrieval methods. Because clients are more concerned with the degree to which web services can deliver the required functionality, discovering Web services using security attributes combined with keyword based methods becomes the natural solution. Hence, quality of protection discovery can articulate proper service queries while providing an overall assessment for web services in delivering the required functionality.

In this paper we provide a methodology based criteria search for web service discovery on quantitative evaluation of security. It helps to select the most suitable security configuration to a consumer businesses process and different levels of trust between services available in service registries. The proposed methodology discovers web services to desirable security requirements useful for service registries with one understandable by consumer. It will allow the service providers easily recalculate his quality of protection if a security or trust level has been changed.

2. RELEATED SURVEY

Web service discovery is a key component in service oriented computing. However, without Web service quality standards, the trustworthiness of business to business interactions can't be guaranteed. In order to provide the trustworthiness, we planned to analyze and make ways to regulate quality of service with the concept of security agent. Ideally, we would extend existing security standards to support various issues for Web services and potentially regulate service protection requirements. Furthermore, without integrating QoP information into the discovery process, current discovery methods might not achieve SOA's strategic goals.

Web services security become so widely exposed that any existing security vulnerability will probably be uncovered and exploited by hackers. To prevent vulnerabilities, developers should apply best coding practices, perform security reviews of the code, execute penetration tests, use code vulnerability analyzers, etc. However, many times developers focus on the implementation of functionalities and satisfying the user's requirements and disregard security aspects. Additionally, numerous developers are not specialized on security and the common time-to-market constraints limit an in depth test for security vulnerabilities. Security vulnerabilities like SQL Injection and XPath Injection seem particularly relevant in web services as they are directly related to the way the web service code is structured[4].Basically, SQL Injection and XPath Injection attacks take advantage of improper coded applications to change SQL commands that are sent to the database or tamper XPath queries used to access parts of an XML document.

The crucial point in web services security negotiation is the identification of metrics which describe the level of protection. The problem, however, is that service providers' QoS in security claims might be trustworthy. Furthermore, letting service providers submit their own QoS to claims the service registries provides room for manipulation. The important issue is that a consumer and a service registry have different views on how these QoS requirements should look like. Service providers might significantly influence how these claims are generated and think of ways to improve them. For that, we need a method to automate the process of measuring QoS in security for registered Web services. Current UDDI registries don't have built-in capabilities to validate or monitor published web services; they include only metadata about businesses and their related Web services. Even if UDDI registries let service provider publish their QoS claims, they could publish false or inaccurate information, or the published information could become passive or not properly configured.

Clients should be able to obtain web service information based on QoS metrics (response time, throughput, and latency) possibly from a trusted service broker. In addition, clients selectively control and manage their search criteria based on QoS attributes in security like confidentiality, integrity and logging ect. would yield more relevant results. Quality of protection in web service discovery properly articulate the service query, it helps the clients to make accurate decisions. For example, a developer who wishes to choose a service have various implementations of a web service metrics might be influenced by analyzing functional and non-functional mechanism deployed on web service. Current UDDI registries and search engines don't allow quality of protection based searching. The ability to search for web services based on quality of protection can greatly influence secure web service discovery operations. Any organization should therefore realize the potential risks which can arise if proper counter measures are not implemented. Therefore, it is important to qualitatively and quantitatively evaluate vulnerabilities of web service to ensure that the system is protected against risks which can reflects web services security reasonably and directly, and it will be convenient for user to choose a best service.

3. SECURITY CLASSIFICATION

To create a classification of security criteria, we consider a model for web services security, as given by a combination of the services presented in [Web Services Security Testing Framework 2002, Colin Wong Daniel Grzelak] [6]. We have found useful to divide metrics into seven types and these security services that must be enforced in order to create a secure environment are listed in the following

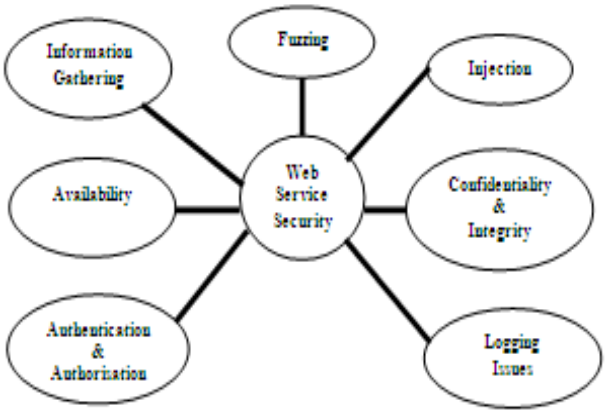

Fig 1: Threat Model for Web Services Security

• ***information gathering*** to identify available web services and obtain detailed information about their operating environment and infrastructure

• ***fuzzing*** to ensure automated testing technique for identifying vulnerabilities in data handling

• ***injection*** involves the subversion of a subsystem to perform actions beneficial to the attacker
• ***confidentiality*** to ensure that data stored in databases and transmitted over a network, cannot be read by unauthorized third parties
• ***integrity*** to ensure that data stored in databases and transmitted over a network cannot be changed by unauthorized third parties
• ***logging*** to ensure that data allows injecting special inputs, overflowing logs, and enumerating and analysing actions logged
• ***authentication*** to ensure proper verification during the log-on process
• ***authorization*** (logical access control) to ensure that the user only has access to that service which is relevant to him/her, and not to other, and
• ***availability*** to ensure that data is available to authorized parties at all times

For introducing a rating corresponding to security level of threats and vulnerabilities, Refer to {Information security technology information safety risk assessment standard}in China, vulnerabilities will be evaluated quantitatively based on attack difficulty, prevalence and harmfulness. Vulnerabilities for Web Services which mainly refer to the protection mechanisms have been used in the message level security, or what kind of security mechanism be deployed. First we try to identify the testing framework for web services security. It aims to provide a guide who wish to adopt a standardised process for evaluating security mechanisms in web services [6].

An experimental evaluation of security vulnerabilities in 300 publicly available web services has presented. Vulnerability scanners have been used to identify security flaws in web services implementations. A large number of vulnerabilities has been observed (177), which confirms that many services are deployed without proper security testing [5][7]. Acunetix Web Vulnerability Scanner used to detected vulnerabilities for each web services to evaluate the security level for each threat that occurs [8]. The assigned values depend strongly on the current technology used for the corresponding criterion, and therefore, represent only a first estimation. The values next to the itemized criteria represent the security estimation used for computing the security rate. The values reach from 1 (absolutely insecure) to 0 (absolutely secure).

Table 1. Threats and Vulnerabilities

Weight	Threat
Information Gathering	
- 0.3	WSDL Retrieval
- 0.4	Soap Error Leakage
- 0.2	Web Method Enumeration
Fuzzing	
- 0.3	Numerical Values
- 0.2	Base64 Encoded Values
- 0.2	Character Strings
- 0.7	General Values
- 0.4	Sub-system Parameters
- 0.4	Output Values
- 0.3	Addressing Parameters
- 0.7	Tokens
- 0.2	Format String Parameters
- 0.6	Logging Values
- 0.4	File Names
Injection	
- 0.2	SQL Injection
- 0.6	Command Injection
- 0.3	LDAP Injection
- 0.5	XPath Injection
- 0.4	Code Injection
- 0.3	XML Special Characters
- 0.6	XML CDATA Sections
Confidentiality & Integrity	
- 0.3	Cipher Choice
- 0.2	Encryption Coverage
- 0.2	Replay Attacks
- 0.4	Integrity Check Coverage
- 0.5	Invalid XML
- 0.7	XML Canonicalisation
- 0.8	Unsupported Algorithms
- 0.5	Failed Policy Requirements
Logging	
- 0.5	Separator Injection
- 0.3	White Space Injection
- 0.6	XML Injection
- 0.5	HTML Injection
- 0.3	New Line Injection
- 0.2	Size Overflow
- 0.4	Information Disclosure
- 0.7	Alternate Data Streams
- 0.5	Not Logged Actions
- 0.7	Logic Flaws
Authentication and Authorisation	
- 0.8	Brute-force and Dictionary Attacks
- 0.4	Forged Credentials
- 0.4	Missing Credentials
- 0.6	Replay Attacks
- 0.3	Authentication Exchange Tampering
- 0.5	Man-in-the-Middle Attacks
- 0.4	Factors of Authentication
- 0.4	Authentication Session Manipulation
- 0.3	Storage of Authentication Credentials
- 0.6	Confidentiality of Authentication
- 0.7	Certificate Verification
- 0.3	ACL and Role Consistency
- 0.5	Token Forgery
- 0.6	Hijacking Attacks
- 0.7	Manipulation across Trust Boundaries
- 0.8	Attacks on Address Filtering
- 0.4	Temporary Files
Availability	
- 0.4	Parameter Tampering
- 0.6	Coercive Parsing
- 0.5	Recursive SOAP
- 0.4	Overly Large SOAP
- 0.4	Entity References
- 0.3	Schema Poisoning
- 0.3	Routing Detours
- 0.5	Transform Attacks
- 0.7	Authentication Flooding

For the presentation of the security in web services our measure computes a value between 1 and 0 applying the catalogue of security criteria. If the interval between 0 and 1 is partitioned in different subintervals, i.e., classes of security, then the computation of the security rating can be compared to statistical pattern matching. The inputs used for the

computation are the tested criteria. The output, i.e., result, is a single value that can be directly matched to a security class that belongs to with some probability.

4. QUALITY OF PROTECTION

We propose a QoP model that covers the quantative evaluation for web services security. Web services are classified into three categories according to the views that they concern. They are service level view, system level view and business level view. The service level view carries on the qualities that can be measured in the service level, i.e. the qualities in use. The system level qualities are the external operational qualities of the service as a system. In this class, there are interoperability, manageability, business processing and security. The factors in the business level cannot be measured or tested, but are calculated or added to the services as meta information that directly affects the business decision.

4.1 Requesters and Providers

The purpose of a Web service is to provide some functionality on behalf of its owner -- a person or organization, such as a business or an individual. The provider entity is the person or organization that provides an appropriate agent to implement a particular service. A requester entity is a person or organization that wishes to make use of a provider entity's Web service. It will use a requester agent to exchange messages with the provider entity's provider agent. In most cases, the requester agent is the one to initiate this message exchange, though not always. Nonetheless, for consistency we still use the term requester agent for the agent that interacts with the provider agent, even in cases when the provider agent actually initiates the exchange. The problem is that a client and a service provider have different views on how these security requirements should look like. We propose a methodology which binds these views and describes a process for selection the security configuration that helps to achieve negotiated level of protection.

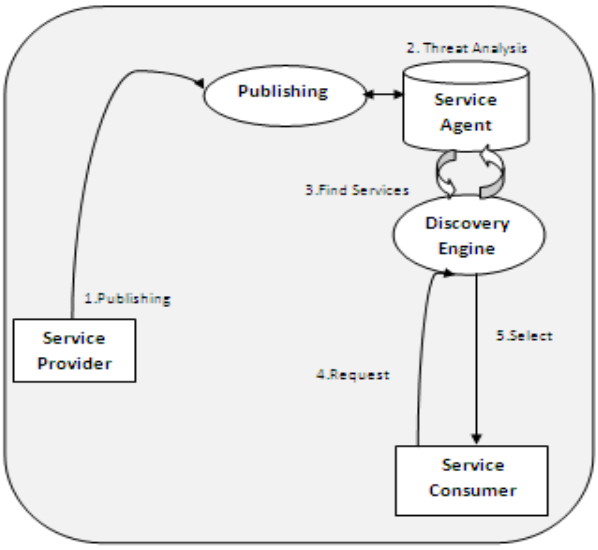

Fig 2: Web Service Security Process

4.2 Service Agent

Our solution to expedite a service discovery process by having service rating done at publication time is called a secure service agent. Instead of considering a service registry as a set of published services, we build a well-organized service knowledge base, which we call a service agent. The service agent consists of a set of service populations and their relationships in rating. A service population is a set of services in which all services are in terms of their input and output parameters. Each service in a population has both WSDL and WSS descriptions.

4.3 Discovery Engine

Seven events, shown in Fig.1, occur for a consumer's request to be matched with an appropriate service. Service providers first publish their service descriptions to a service agent publishing engine. Then service consumers request that the security agent provide a security service adequate to their needs. The discovery mechanism finds services matching the request. The discovery engine queries the service agent in order to answer the find operation. If the discovery engine finds a set of matched secure services, the engine selects the best from the candidate set. The discovery engine injects the newly discovered service into the agent community for reuse and notifies the service consumer. This paper focuses on the discovery engine of Figure 2. The engine fields a request from consumers, parses it, invokes find on the matchmaking mechanism, and uses the results to obtain an executable service that is then returned to the consumer.

5. RATING

With the increasing number of Web services available in individual domains of interest, complexity and market -need problems specific to service development and invocation can arise. To address these problems, organizations should increase management control over security service implementation. Web service evaluation can support management control by assessing Web service offerings according to specific security guidelines. Web service evaluation helps fill the gap between service features and client requirements. Evaluating web services security can help service providers to determine the required level of security; it doesn't necessarily provide clients with trustworthy evaluations. A trusted Web service broker that evaluates web services security could provide clients with trustworthy information.

5.1 Service Discovery

All services in the populations match the consumer's request depend only on functional requirements. Non-functional requirements must be examined to refine the new solutions. We use a Quality of protection ontology to define these security metrics. The currently defined set of QoP criteria appears in Table1. Since the consumer's request can contain functional and other non-functional attributes, the match making algorithm uses concepts to select the best service from a registries. Here, ranks are assigned to each service using the metrics driven approach. The rank of each individual ws1,ws2,ws3...wsn is calculated as the number of individuals criteria occurs. Once rank has been calculated for each service in a registries. In the case of a ws-n-way tie, service can be chosen randomly. QoP data for the services are then aggregated and compared to those provided by the consumer attributes, to determine if the composed service will meet their request.

Algorithm

Input: User request with specified meta search
Output: Set of secure services from registries
R(h): Select all the services which matches the functional requirements of user from registries.
Let *R(h)={ws1,ws2.....wsn}*

wss (h): Choose the set of services from registries with security evaluation.
Let *wss(h)={ws1(s), ws2(s), ….wsi(s)}*
Step 1 : For each web services wsi in u(h)
//find the services that match the QOS requirements
Step 2: QoS Selection=Qos_Match (u(h) ,WS);
Step 3: If wss(h) specified then
Security_Match (QoS,wss(h)) defined
Step4: If wss(h) ratings found then
//find the services that matches security criteria
Step5: return output of available services in wssi in R (h) according to criteria rank}
Step6 :{ Else return the output of available services wsi in R (h)}

5.2 Evaluation

The complexity for estimating the effectiveness of web service security increases if it involves all the sixty five parameters [6]. As a solution to this, we follow the mechanism of heuristic to evaluate the parameters and estimate the criteria rank for each and every parameter. The central theme of heuristic is the intelligent use of heuristic function i.e. prioritized weights assigned to each parameter to achieve the maximum protection in the final optimized result. We split the entire set of security parameters into different criteria and then evaluate the overall rank using heuristic function i.e prioritized weights assigned to each parameter to achieve maximum protection in the secure web service discovery. For involving a minimum number of chosen parameters falling under every factor with high priority, that decide and evaluate the rank of web service security.

Step 1:.Input the list of web service from UDDI registries. This set includes WS1,WS2.….,WSn that may be exploited to various security vulnerabilities assets. Each vulnerability is identified by a unique ID and named correspondingly.

Step 2: Each threat and vulnerabilities is identified by a name and unique ID {WV1, WV2, WV3…..,WVNn}. Each web service is assigned with estimated threat probability (TP). The TP values may range between 0 and 1 where 0 means that threat occur and 1 means that it will neutralise.

Step 3: Calculating the total value of each assets. Value of assets is calculated by summing value of all assets in the given testing framework mechanism. ValSA – Value of all system assets.

$$ValSA = \sum Val(Ai) \text{ where i runs over all assets}$$

Step 4: Calculate the overall value of the assets by taking all the assets final value and perform summing operation on all the assets by defined r value.

Occurrences – no. of attacks examined
Gp – Value of assets X No.of occurrences

5.2.1 Risk Calculation

The following tables show the calculation of quality of protection (QOP) using heuristics approach. Here predefined vulnerability attack has been chosen and perform security audit using Acunetix Web Vulnerability Scanner [8] an automated web application security testing tool that exploit hacking vulnerabilities. After scan for common application vulnerabilities in each web services to detect number of times attack occurs. In order to evaluate the attacks we identify number of occurrence of attacks that leads to major security flaws.

Information Gathering GP = (Number of attacks occurence)

Overall GP = Σ (Values of all the assets parameter)

Table 2. Information Gathering

Web Services	WSDL Retrieval Occurrences	WSDL Retrieval GP	Web Method Enumeration Occurrences	Web Method Enumeration GP	Soap Error Leakage Occurrences	Soap Error Leakage GP	Overall Rating	Criteria Rank
Dictionary Service	5	1.5	6	1.2	4	1.6	4.3	4
Error Mailer	4	1.2	4	0.8	5	2.0	4.0	1
Get Joke	2	0.6	5	1.0	11	4.4	6.0	7
Fast Weather	5	1.5	12	2.4	2	0.8	4.7	5
My Service	7	2.1	15	3.0	4	1.6	6.7	9
Currency Converter	8	2.4	12	2.4	4	1.6	6.4	8
XigniteNews	11	3.3	4	0.8	2	0.8	4.9	6
Email Validation	2	0.6	6	1.2	6	2.4	4.2	3
Alexa Web Search	7	2.1	8	1.6	1	0.4	4.1	2

Information GatheringGP =Attack ocurrence(WSDL+WebMet+Soap)
(Total number of assets)
WSDL = Attack Occurrence * Threat_Value
= (5 * 0.3) = 1.5
Web Meth. = Attack Occurrence * Threat_Value
= (6 * 0.2) = 1.2
Soap = Attack Occurrence * Threat_Value
= (4 * 0.0.4) = 1.6
Information Gathering GP = (1.5+1.2+1.6) = 4.3 GP

Table 3. Fuzzing

Web Services	Base64 Encoded Values Occurrences	Base64 Encoded Values GP	Subsystem Parameters Occurrences	Subsystem Parameters GP	Format String Parameters Occurrences	Format String Parameters GP	Tokens Occurrences	Tokens GP	Overall Rating	Criteria Rank
Dictionary Service	4	0.8	5	2.0	3	0.6	2	1.4	4.8	1
Error Mailer	2	0.4	3	1.2	1	0.2	5	3.5	5.3	4
Get Joke	3	0.6	2	0.8	5	1.0	4	2.8	5.2	3
Fast Weather	5	1.0	3	1.2	7	1.4	8	5.6	9.2	8
My Service	7	1.4	4	1.6	3	0.6	6	4.2	7.8	7
Currency Converter	2	0.4	6	2.4	8	1.6	1	0.7	5.1	2
XigniteNews	5	1.0	6	2.4	3	0.6	5	3.5	7.5	6
Email Validation	8	1.6	8	3.2	9	1.8	7	4.9	11.5	9
Alexa Web Search	9	1.8	5	2.0	2	0.4	3	2.1	6.3	5

Table 4. Injection

Web Services	LDAP Injection Occurrences	LDAP Injection GP	XPath Injection Occurrences	XPath Injection GP	Command Injection Occurrences	Command Injection GP	XML Special Characters Occurrences	XML Special Characters GP	Overall Rating	Criteria Rank
Dictionary Service	3	0.9	3	1.5	9	5.4	5	1.5	9.3	9
Error Mailer	4	1.2	2	1.0	2	1.2	2	0.6	4.0	1
Get Joke	1	0.3	6	3.0	6	3.6	6	1.8	5.7	2
Fast Weather	6	1.8	4	2.0	7	4.2	2	0.6	8.6	6
My Service	8	2.4	3	1.5	4	2.4	9	2.7	9.0	8
Currency Converter	3	0.9	2	1.0	5	3.0	5	1.5	6.4	4
XigniteNews	5	1.5	1	0.5	6	3.6	1	0.3	5.9	3
Email Validation	2	0.6	8	4.0	4	2.4	3	0.9	7.9	5
Alexa Web Search	1	0.3	9	4.5	3	1.8	7	2.1	8.7	7

Table 5. Logging

Web Services	Separator Injection Occurrences	Separator Injection GP	XML Injection Occurrences	XML Injection GP	New Line Injection Occurrences	New Line Injection GP	SizeOverflow Occurrences	SizeOverflow GP	Overall Rating	Criteria Rank
Dictionary Service	4	2.0	5	3.0	4	1.2	3	0.6	6.8	4
Error Mailer	2	1.0	2	1.2	6	1.8	6	1.2	5.2	1
Get Joke	4	2.0	6	3.6	3	0.9	2	0.4	6.9	5
Fast Weather	6	3.0	3	1.8	2	0.6	3	0.6	6.0	2
My Service	1	0.5	5	3.0	9	2.7	5	1.0	7.2	6
Currency Converter	8	4.0	1	0.6	6	1.8	1	0.2	6.6	3
XigniteNews	5	2.5	8	4.8	3	0.9	8	1.6	9.8	9
Email Validation	7	3.5	7	4.2	1	0.3	5	1.0	9.0	8
Alexa Web Search	5	2.5	6	3.6	5	1.5	2	0.4	8.0	7

Table 6. Availability

Web Services	Coercive Parsing		Recursive SOAP		Routing Detours		Authentication Flooding		Overall Rating	Criteria Rank
	Occurrences	CF	Occurrences	CF	Occurrences	CF	Occurrences	CF		
Dictionary Service	3	1.8	2	1.0	6	1.8	3	2.1	6.5	1
Error Mailer	4	2.4	8	4.0	2	0.6	2	1.4	8.4	9
Get Joke	1	0.6	2	1.0	3	0.9	7	4.9	7.4	4
Fast Weather	3	1.8	5	2.5	1	0.3	5	3.5	8.1	8
MyService	4	2.4	1	0.5	6	1.8	3	2.1	6.8	2
CurrencyConverter	2	1.2	6	3.0	5	1.5	2	1.4	7.1	3
XigniteNews	3	1.8	7	3.5	4	1.2	2	1.4	7.9	6
EmailValidation	4	2.4	4	2.0	5	1.5	3	2.1	8.0	7
AlexaWebSearch	3	1.8	8	4.0	3	0.9	1	0.7	7.4	4

Table 7. Confidentiality & Integrity

Web Services	Cipher Choice		Integrity Check Coverage		XML Canonicalisation		Replay Attacks		Overall Rating	Criteria Rank
	Occurrences	CF	Occurrences	CF	Occurrences	CF	Occurrences	CF		
Dictionary Service	2	0.6	4	1.6	3	2.1	2	0.4	4.7	3
Error Mailer	6	1.8	3	1.2	2	1.4	6	1.2	5.6	4
Get Joke	1	0.3	1	0.4	5	3.5	1	0.2	4.4	2
Fast Weather	3	0.9	6	2.4	4	2.8	3	0.6	6.7	5
MyService	5	1.5	5	2.0	7	4.9	6	1.2	9.6	9
CurrencyConverter	4	1.2	3	1.2	2	1.4	4	0.8	8.0	8
XigniteNews	1	0.3	2	0.8	1	0.7	3	0.6	2.4	1
EmailValidation	7	1.4	7	2.8	2	1.4	9	1.8	7.4	7
AlexaWebSearch	2	0.6	6	2.4	3	2.1	8	1.6	6.7	5

Table 8. Authentication & Authorisation

Web Services	Forged Credentials		Missing Credentials		Token Forgery		Certificate Verification		Overall Rating	Criteria Rank
	Occurrences	CF	Occurrences	CF	Occurrences	CF	Occurrences	CF		
Dictionary Service	3	1.2	5	2.0	6	3.0	5	3.5	9.7	7
Error Mailer	5	2.0	4	1.6	3	1.5	4	2.8	7.8	5
Get Joke	6	2.4	2	0.8	2	1.0	2	1.4	5.6	2
Fast Weather	4	1.6	7	2.8	8	4.0	7	4.9	13.3	9
MyService	2	0.8	1	0.4	4	2.0	4	2.8	6.0	3
CurrencyConverter	3	1.2	3	1.2	3	1.5	1	0.7	4.6	1
XigniteNews	4	1.6	6	2.4	1	0.5	6	4.2	8.7	6
EmailValidation	8	3.2	4	1.6	7	3.5	2	1.4	9.7	7
AlexaWebSearch	1	0.4	3	1.2	2	1.0	6	4.2	6.8	4

Table 9. Overall Ranking

Web Services	Information Gathering	Fuzzing	Injection	Logging	Availability	Confidentiality & Integrity	Authentication & Authorisation	Overall Rating %	Criteria Rank
Dictionary Service	4.3	4.8	9.3	6.8	6.5	4.7	9.7	46.1	4
Error Mailer	4.0	5.3	4.0	5.2	8.4	5.6	7.8	40.3	1
Get Joke	6.0	5.2	5.7	6.9	7.4	4.4	5.6	41.2	2
Fast Weather	4.7	9.2	8.6	6.0	8.1	6.7	13.3	56.6	8
MyService	6.7	7.8	9.0	7.2	6.8	9.6	6.0	53.1	7
CurrencyConverter	6.4	5.1	6.4	6.6	7.1	8.0	4.6	44.2	3
XigniteNews	4.9	7.5	5.9	9.8	7.9	2.4	8.7	47.1	5
EmailValidation	4.2	11.5	7.9	9.0	8.0	7.4	9.7	57.7	9
AlexaWebSearch	4.1	6.3	8.7	8.0	7.4	6.7	6.8	48	6

6. CONCLUSION

Automated secure web services discovery have long been a goal of researchers. With an objective to provide protection in web service discovery, we identified several security parameters classified under different criteria to evaluate web services security. The central theme of heuristic approach is to achieve the maximum protection in the final optimized result. Ranking of web services done through heuristic approach where in the parameters are assigned prioritized threat value which meliorates the evaluation of quality of protection. It is both time-consuming and error-prone to manually select services based on parameter chosen only. We present a model which is of great help in rating the web services with the end user's perspective basis on different levels of views. This work reduces search time in web services discovery and future by extending new vulnerabilities possible in web services security.

7. REFERENCES

[1] H. F. Tipton and M. Krause, Information Security Management Handbook, CRC Press, 2004.

[2] J.Mirkovic,D-WARD: Source-End Defense Against Distributed Denial-of-service Attacks, Ph.D.Thesis, University of California, LosAngeles, 2003.Handbook, CRC Press, 2004.

[3] Al-Masri, E.; Mahmoud, Q.H.; Towards Quality-Driven web service Discovery, P u b l i s h e d by the IEEE Computer Society, IT Pro May/ June 2008.

[4] Spyrost. halkidis, Alexander chatzigeorgiou, George stephanides, "A Practical Evaluation of Security Patterns", Math. Subjects Classification 2000: 94A60, 14G50, 68Q99.

[5] DuanYouxiang1 and Gao Yang. "Evaluating Vulnerabilities Quantitatively Based On the Rank of Web Services Confidentiality", Journal of Next Generation Information Technology, volume 2, Number 1, February, 2011.

[6] Colin Wong and Daniel Grzelak, "A Web Services Security Testing Framework", SIFT SPECIAL PUBLICATION, Information security services, Version 1.00.

[7] Marco Vieira,Nuno Antunes, and Henrique Madeira "Using Web Security Scanners to Detect Vulnerabilities in Web Services". IEEE/IFIP Intl Conf. on Dependable Systems and Networks, DSN 2009,Lisbon, Portugal, June 2009.

[8] Acunetix Web Vulnerability Scanner, 2008, http://www.acunetix.com/vulnerability-scanner/

[9] John Steven and Gunnar Peterson,"A Metrics Framework to Drive Application Security Improvement", IEEE Security & Privacy, vol. 1, no.4, 2003, pp.88–91. H. F. Tipton and M. Krause, Information Security Management Handbook, CRC Press, 2004.

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

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

[12] Artsiom and Yautsiukhin, "Quality of Protection Determination for Web Services". http://bis.kie.ae.poznan.pl/10th_bis/wsiqs1.pdf

[13] Bachar Alrouh and Gheorghita Ghinea, "A Performance Evaluation of Security Mechanisms for Web services", 2009 Fifth International Conference on Information Assurance and Security.

[14] Alain Geroges Vouffo Feudjio,"Availability Testing for Web Services", ISSN 0085-7130 © Telenor ASA 2009.

[15] D. J. Mandell and S. A. McIlraith. A Bottom-Up Approach to Automating Web Service Discovery, Customization, and Semantic Translation. In the Proceedings of the Twelfth International World Wide Web Conference Workshop on E-Services and the Semantic Web (ESSW'03), Budapest, Hungary, 2003.