Restful web services security by using ASP.NET web API MVC based

Muhammad Imran Hussain and Naveed Dilber
Department of MS Computing (SZABIST) Karachi Pakistan.
muhammad.imran@live.com and naveed.dilber@szabist.edu.pk

***Abstract*- The use of web services APIs play a vital role in the development of project in web where as in the era of rapid development; the use of web services are involved as a software as services; with the assistance of Internet web sites containing the full of useful information. Therefore, the trend of REST has increases, the uses of services development gained much more attention and widely used in last few years. Hence, the values of information security are becoming a challenge. Similarly, the restful web services are becoming a favorite, because it is lightweight, easy to implement and it also compatible with the cross platform solutions like mobile, PDAs and enterprises applications. As it has not any predefined security model, therefore the engineers also take the responsibility to implement the techniques to secure the information of enterprises. In this study our focus is to explore the best practices such as claim based, web tokens and delegation based by using Asp.Net Web API.**

Keywords: Restful APIs; REST; Web services security.

I.INTRODUCTION

In past, most of the companies effectively assured that the information security via operation system deals with controls access on resources such as files and network connections. However, the most of the attacks may occur at higher levels of abstraction so that it can target the internal behavior of the system or application [1].

The enterprises are moving quickly towards the development of decentralized, flexible and layered application architecture (E.g SOA) which can gives the solutions to the clients on cross platform environment such as web or mobile plate formed supported. In order to make the transition, the treatment of application level of security consider as a major threat [2-4].

The security of application has become more critical sine the emergence of web 2.0. It is just because of adoption of ease and openness, there has been an explosion of publics API's which allows the developers to call the functions and its data from multiple diverse services to create the new applications [5].The Representational State Transfer (REST) based web services such as Google geocoding and Facebook APIs are increasingly popular today. Not surprisingly, the application developers have begun to take advantage of this technology to facilitate business processes in the enterprises. In the business environment of today's business processes are more likely to be dynamic, distributed and collaborative, so it is necessary to adapt business processes more often, and process integration through organizational boundaries with partners. In addition, organizations also need to exhibit critical business processes, embedded in millions of lines of code from mainframe developed in the last four decades (Mitchell, 2006), and integrate these features with new interfaces or web-based user mobile. The REST technology provides a cost effective and efficient way to support the integration of business processes so tight alternative.

A. Technology Background

The principle of rest overlooks specifies the execution of the module or section and protocol syntax for the purpose to emphasis on the distinctiveness of the components, restrictions on its communication with others modules and their interpretation of pertinent data [1-2]. This is because this architectural style is applied behind the implementation of restful web services that focuses on system resources. It architectural patterns obey the rules of the W3C's web architecture and empower it design principles of the web flavors, to developing the intensity on the established infrastructure of the Web. It utilizes the semantics of HTTP whenever possible and most of the principles, constraints, and best practices published by W3C's Technical Architecture Group (TAG) also apply. Therefore the rest based architecture relates with the service oriented architecture (SOA).This restricts HTTP interfaces with the four types of verbs such as POST, PUT, GET and DELETE. The data format of rest a support not only XML format but is also give the flexibility in JSON (Java script object Notation) data format for messaging. Therefore, its implementation is based on resource oriented view of the world rather than the view of object oriented as many developers are familiar with. The word resource uses as abstraction l of the things which are recognized as a URIs. In order to perform the REST, it is developed with HTTP and XML or JSON. Therefore these two key terms are generally significant for the development of restful web services which are known as identification and resources. The most import thing here to understand is that the REST encourages to the use of HTTP methods, these methods supports all browsers.

B .Research Methodology

The empirical research will be taken to secure the rest API principles and best practices.

C. Problem Statement

As the major problem of restful web services is security concerns, because it does not have own security model or predefined security methods. The objective of this study is to

explore ASP.NET web APIs that will be used MVC framework to implement the restful web service; the use of different security techniques such as claim based, OAuth2 and token simple web token and delegation based as it has serious security constraints as it is cashable, stateless, client-server etc. Based on the study the best practices will be identified to secure the representation of the resource of information.

II.WEB SERVICES CONCEPTS & ARCHITECTURE

A. Traditional Web services

Essentially web services are built on the procedure of service oriented architecture (SOA), in which the distributions of the software systems consider as a set of services. Therefore, to make the accessibility with another application, there should be mechanism that can provide the service invoker discovery. Figure 1 illustrates the three basis components interaction between SOA.

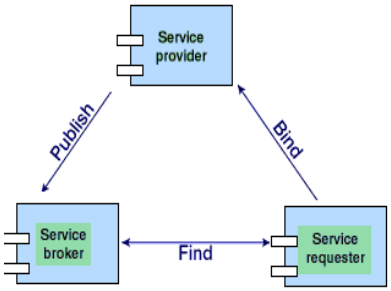

Fig. 1. Component of SOA [18]

Essentially the web services carry out three basic and significant functions the first one published, the second is finding and the third is binding. This become dependability of the service supplier to publish the services for services broker service, a service requester who needs to consume these services into application, the service Broker application and then eventually unite with the service provider this way has been completed the entire process flow of the web services[19].

The technologies that support these three basic functions are UDDI, WSDL and SOAP. Here our focus will on SOAP.

1-SOAP

It is basically used as a protocol which is used to for exchanging the information in the development of web services. The web services that hold SOAP messages have flexibility to develop in any programming language in any platform, but this is most important it can be consumed by any other enterprise application regardless of the development in programming language. SOAP provides an entity which uses XML which is used as a data carrier purpose [20].

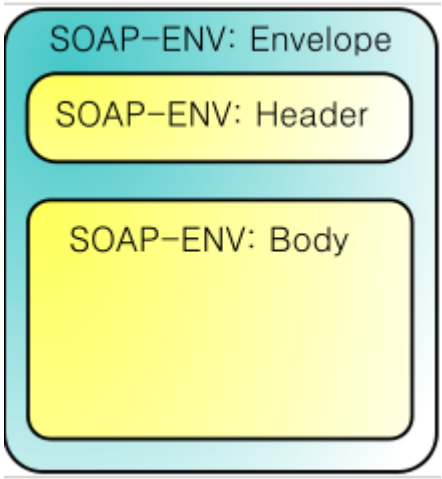

Fig. 2. SOAP Message Structures [20]

B. Restful web Services

The concepts of the REST illustrate a set of architectural principles in which the design of web services can be implemented. The HTTP protocol is used for the transmission of data between client and server. The implementation of REST is based on four basic design principles is also known as constraints are as follows:

- The use of resource reorganization
- The manipulation and representation of each resource
- The Self-description of messages.
- The use of HTTP as the engine of the application state.

These limitations define interactions and architecture that integrates the web for what reason, the cornerstone of the web is known as resources. A resource can be something such as file, script, collection of resources) named as a target hypertext. After getting the response from the resource, the representation of the source received from the client, which may have a different property preparing the server resource. The manipulation of the resources identified with messages giving identification is standard meanings; anywhere in the web, messages are HTTP methods.

In SOAP based and REST based web services the management of state is significant; the fourth principles exhibits that any client or server state is reserved in hyper media in which it may swap the information or messages via links or universal resource identification (URI). Any information between client and server is based on the ground state to send messages, but it keeps them state less. It is ease to verify any design against such a simple description. Any inconsistency will be easy to identify. In reply to a request for a resource, the client receives a depiction of that resource, which can have an alternate format to see the property of the server resources. Resources are handled through messages that have a standard meaning; Web, these messages are HTTP

methods. The constraints of REST are client/server, stateless, caching, uniform interface, resource identification, self-descriptive messages and manipulation of resources through representations, layered system and here we will discuss only code on demands.

1-Code on demand

The server can extend the functionality of the client by passing executable code. For example, a server can send JavaScript to the client so that it can do some type of operation on the data. If we read these principles carefully, we note that their primary focus is scalability. The fact that the server should not store client information permits it to save memory. The layered system permits us to use cache servers as a load balancer to obtain scalability. Adding new servers while adhering to the client-server principles allows us to change the implementation (for example, going from a SQL database to No SQL storage) without the client's knowledge. The web itself is based on REST because of the URL identification. When the type the URL in the browser to get a representation in the HTML format, and we use a link to the transfer the state to another page. Another aspect of REST is that it contrast with the SOAP (RPC) in which the operation performed on the basis of resource, the HTTP verb are used that makes the combination with URL whereas the HTTP has the notation of verbs to used GET and POST the data for manipulation since the majority of the most support these two verbs, but others are also mentioned in HTTP (RFC 2616) that can be used for the other operation such as GET, PUT, POST, DELETE, TRACE etc. When the server is issued by the server, the server takes the responsibility to parse and build the response which have return the data or result which is received from the client. Every response represented with a state, and HTTP status should be semantically used to inform the client result. There are several types of HTTP status codes such as Information, success, redirection, client errors and server's errors.

III-FRAMEWORK & IMPLEMENTATION

The consumption and utilization can be used in wide range, such as client's browser, mobiles, iPhone and tables etc. As it is light weight and it use the MVC framework. It builds the HTTP services with easy and simple way. Web API is an open source and it is an ideal platform to develop the Restful web services over dot net platform. There is not any doubt; the role of technology enhancement makes the life easy and conformable. So that the movement from the web to the application world. This is because it became the need that modern devices application can expose the data in efficient manner; there should an API which can be compatible with brewers and all these devices.

The API of the web is the framework for exposure and service data to devices. Besides Web API is an open source ideal for building a restful web service in dot net platform. Use the full features of HTTP and URI, request header response caching, versioning and different forms of content formats

A. Framework

In this study our focus on the MVC (Model-View-Control) framework, the benefits of this framework is that it decouple the presentation layer and business logic to become a de-facto blueprint, that the developers use to designer and implement flexible, scalable and maintainable the web solutions. Therefore, the modern enterprise system frequently depends on MVC frame work implementation on the internet or internet. MVC is composed on three components.

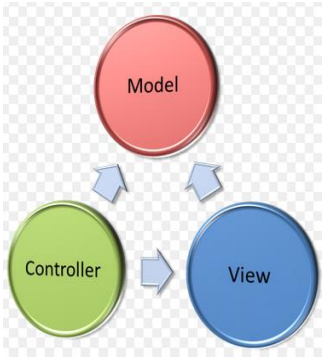

Fig. 3. Model View Controllers [7]

B. Web API Architecture

Under Microsoft ASP.Net framework; the Web API makes it easy to build HTTP services. The traveling of messages starts from server to client with the help of HTTP Request and HTTP Response flow. The diagram shows the Web API message life cycle. The below section we will also look at the various extensible points in the WEB API pipe line.

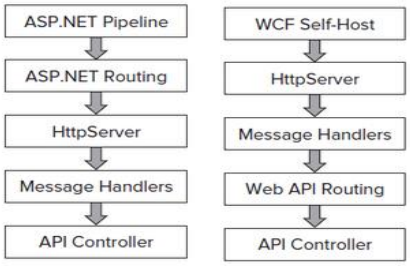

Fig. 4. The Web API Pipeline [8]

C. Web API Hosting

There are two ways to host the WEB-API.1) ASP.Net hosting 2) windows service it is also known as self-hosting.

D. Http Message Handlers

The Message Handler is the class which is used for sends and receives HTTP request and response. This is derived with abstract HTTP message handler class. Typically, it is

collections of message handlers are chained together. The first handler receives HTTP request, after some processing, the request goes back to the next handler. Similarly the response collection has to be created, it goes back up the chain such type of pattern is known as delegating handler [9].

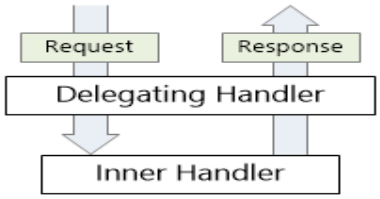

Fig. 5. Http Message Handlers [9]

The routing configuration in Web-API is slightly different form ASP.Net MVC routing. The web API uses the Http Route Collection because it can reuse the routing of Asp.Net MVC. This is the only reason it is different; while in Web-API, it will not only register the HTTP route object but it also create a wrapper Route object while creating the restful web services API project in Asp.Net MVC.

```
        }

        string encodedCredentials = authHeader.Parameter;
        byte[] credentialBytes = Convert.FromBase64String(encodedCredentials);
        string[] credentials = Encoding.ASCII
                                    .GetString(credentialBytes).Split(':');

         if (!_service.Authenticate(credentials[0], credentials[1]))
         {
             return Unauthorized(request);
         }

         string[] roles = null; // TODO
         IIdentity identity = new GenericIdentity(credentials[0], "Basic");
         IPrincipal user = new GenericPrincipal(identity, roles);
         HttpContext.Current.User = user;


         return base.SendAsync(request, cancellationToken);
     }

    private Task<HttpResponseMessage> Unauthorized(HttpRequestMessage request)
    {
        var response = request.CreateResponse(HttpStatusCode.Unauthorized);
        response.Headers.Add("WWW-Authenticate", "Basic");

        TaskCompletionSource<HttpResponseMessage> task = new TaskCompletionSou
rce<HttpResponseMessage>();
        task.SetResult(response);
        return task.Task;
    }
}
```

IV. SECURITY

As it has discussed earlier section it does not any predefined security model in restful. The Restful web services security is a multi-faceted: it comes securing data, as well as the entire communication. It must protect the confidentiality and integrity of data. Data in transit must be filtered for malicious payload. There should be a mechanism for authentication and access control that is responsible for ensuring the privacy of the participants in communication is not compromised. As compared to the web services security framework, Restful services depends on various add-ons that work on the top of HTTP. Similarly the use of HTTPS is widely used for confidentiality by adopting the message level security header mechanism. HTTP also support the basis and digest based authentication mechanism as there is not any doubt it has some weakness.

A. Authentication & Authorization Approaches

Authentication is the process which is used by the server when the server needs to know who is exactly accessing their information. It is normally used by the client, while authentication is the act of verifying the client request. When the client makes the request, the authentication token in X-Authorization header or token of the query string parameters pass as request, the server verify the presence of authorization token , the checks some validation such as expiry of the token and parse the authentication principle based on the token contents. If the server authorized the client token, the restful web service will be able to continue the further processing. Therefore the authorization process determines the server, the client has the permission to use the resources or not

B. Basic Authentication

It provides a mechanism to validate the request of sender and receiver before processing the HTTP request. It also protect against the denial of service attacks [14]. The credentials are required to read the receipt message which is based on user name and password of the client. Actually it encodes the credentials in base 64 and placed it on header string. To read the message of the receipt, the credential will be validating from the header encoded string certificate, it will process further when the authentication is successful otherwise unauthorized code 401 will be returns.

It uses over on SSL connection (HTTPS) to secure the use name and password of the client to expose other by using base-64 encoded techniques. The internet information service (IIS) also support the basic authentication, the user is authenticated via windows authentication credentials. Therefore, the use must have the credentials of the server domain. For instance, the request using the basis authentication is like:

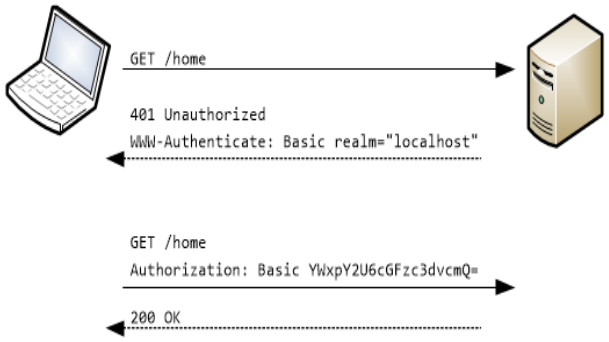

Fig. 6. Basic authentication work flow [15]

```
GET http://localhost:1085/api/Values HTTP/1.1
Host: localhost:1085
Proxy-Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.31 (KHTML,
like Gecko) Chrome/26.0.1410.43 Safari/537.31
Cache-Control: no-cache
Authorization: Basic ZW1hOnB3ZA==
Accept: */*
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8,it;q=0.6
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
```

As the username and password are only base64-encoded in the format of username:password and to avoid passwords are exposed to others, Basic Authentication should be always used over SSL connection (HTTPS). The authorization header with the username and password are encoded in Base64, it is denoted with bold text. To manage the authentication request there should a class of Message Handler that receives the request before it arrives to the controller and decides if the user has the right to access the resource or not. For example

```
public class BasicAuthenticationHandler : DelegatingHandler
{
    private readonly IAuthenticationService _service;

    public BasicAuthenticationHandler(IAuthenticationService service)
    {
        _service = service;
    }

    protected override Task<HttpResponseMessage> SendAsync(
                                HttpRequestMessage request,
                                CancellationToken cancellationToken)
    {
      AuthenticationHeaderValue authHeader = request.Headers.Authorization;
      if (authHeader == null || authHeader.Scheme != "Basic")
      {
          return Unauthorized(request);
      }

      string encodedCredentials = authHeader.Parameter;
      byte[] credentialBytes = Convert.FromBase64String(encodedCredentials);
      string[] credentials = Encoding.ASCII
                                .GetString(credentialBytes).Split(':');

       if (!_service.Authenticate(credentials[0], credentials[1]))
       {
           return Unauthorized(request);
       }

       string[] roles = null; // TODO
       IIdentity identity = new GenericIdentity(credentials[0], "Basic");
       IPrincipal user = new GenericPrincipal(identity, roles);
       HttpContext.Current.User = user;


       return base.SendAsync(request, cancellationToken);
   }

  private Task<HttpResponseMessage> Unauthorized(HttpRequestMessage request)
  {
      var response = request.CreateResponse(HttpStatusCode.Unauthorized);
      response.Headers.Add("WWW-Authenticate", "Basic");

      TaskCompletionSource<HttpResponseMessage> task = new TaskCompletionSou
rce<HttpResponseMessage>();
      task.SetResult(response);
      return task.Task;
   }
}
```

```
AuthenticationHeaderValue authHeader = request.Headers.Authorization;
if (authHeader == null || authHeader.Scheme != "Basic")
{
    return Unauthorized(request);
}

string encodedCredentials = authHeader.Parameter;
byte[] credentialBytes = Convert.FromBase64String(encodedCredentials);
string[] credentials = Encoding.ASCII.GetString(credentialBytes).split(':');
```

```
AuthenticationHeaderValue authHeader = request.Headers.Authorization;
if (authHeader == null || authHeader.Scheme != "Basic")
{
    return Unauthorized(request);
}

string encodedCredentials = authHeader.Parameter;
byte[] credentialBytes = Convert.FromBase64String(encodedCredentials);
string[] credentials = Encoding.ASCII.GetString(credentialBytes).split(':');
```

The class Authentication Service implements Authentication Service interface, if the credentials is true then the client user can access the resources.

```
public interface IAuthenticationService
{
    bool Authenticate(string user, string password);
}

public class AuthenticationService: IAuthenticationService
{
    public bool Authenticate(string user, string password)
    {
        //Do database calls and check if
        //the user and password matches.
        return true;
    }
}
```

This below mentioned method is mean to build the Principal and Identity information, which can be used from the use of ASP.Net membership provider.

```
string[] roles = null; // TODO
IIdentity identity = new GenericIdentity(credentials[0], "Basic");
IPrincipal user = new GenericPrincipal(identity, roles);
HttpContext.Current.User = user;

return base.SendAsync(request, cancellationToken);
```

If the Authentication header is present, it decodes its value from Base64 and extracts the value of the username and password

If the authecation header is present, then it decode it and it get the values of the crentdential of user and its passoword. If it is valid or not exist into it , the unauthrozed message 401 code sent to the client as a message.

```
private Task<HttpResponseMessage> Unauthorized(HttpRequestMessage request)
{
    var response = request.CreateResponse(HttpStatusCode.Unauthorized);
    response.Headers.Add("WWW-Authenticate", "Basic");

    var task = new TaskCompletionSource<HttpResponseMessage>();
    task.SetResult(response);
    return task.Task;
}
```

A. Claim based Authentication

The claim based authentication is more general authentication mechanism that allows the client/ user to authenticate on the basis of external system that provided asking the system with claim about user. In general word the claim is basically the piece of information that describes given identity in some aspects. It takes claims as a name value key pairs which is comprised on authentication token which can have some signature, for understanding purpose it is same as envelop which hold the information about user for claim. The OAuth provide the rich mechanism which is used for external authentication like Face book, twitter, yahoo etc. It is become trend, it can be found in any application like share point 2013. MVC5, WEB API 2 also support claim based authentication.

B. Token based Authentication

Although it is not standard model it is widely used as Google APIs, face book, Amazon web services etc. Often the client is not a user but another application, in which case username and password are not relevant. In this scenario, there should a token which gives the access to the application by using token validation. If the token is authenticated then application will be accessible otherwise unauthorized token and the token validity will also considered. Technically, this process with similar to the basic authentication; generally header authorization techniques can be used to secure the method. While it is not standard, other custom header techniques could be used as well. In the same way the header could by encrypted with private/public key pair to get the better security level.

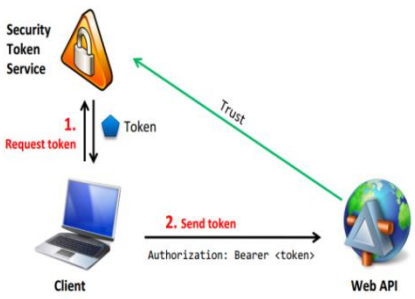

Fig. 7. Token-based security work flow [16]

A. Preventing Cross-Site Request Forgery

In cross site request forgery is also known as one click attack or session riding and it is abbreviated as CSRF[17]. It is an attack in which a malicious site sends a request to a vulnerable sit where the user is currently logged in.

```
<h1>You Are a Winner!</h1>
    <form action="http://example.com/api/account" method="post">
        <input type="hidden" name="Transaction" value="withdraw" />
        <input type="hidden" name="Amount" value="1000000" />
 <input type="submit" value="Click Me"/>
    </form>
```

The above mentioned code, when user click on the form button, the malicious page run a script which sends an AJAX request. Furthermore, CSRF attack does not prevent using SSL authentication, this is because the malicious site can send an https request. Normally such type of attack is possible with the use of cookies authentication. Thus, such type of attacks are not only limited to exploiting the cookies[18]. Basic and digest authentication are also vulnerable. The restful web api provide the mechanism to the prevent from such type of attacks.

B. Secure with SSL

The SSL is one of the suitable way to secure the restful web services. By using encryption techniques, there is a way to achieved the security of data transmission by using HTTPS[10].On the other side the third party authentication is getting more popularity to authorize the application by using OAuth solutions, it supports third party application authentication such as Face book, Google, and twitter[11]. Therefore, the application's user can hold the credential at once and us it whenever the access to the server resources is necessary. It is remarkable that the measures of the security are optimal for the implementation of the security management and likely overlooked when restful web services are built upon adhoc additions to legacy application. In an enterprise system cannot achieved the success without implementation of security. So it must require appropriate protection is supplied when information has be created and it submit as a final decision and gives the protection against unauthorized modification, destruction, loss or discourse of information[12]. It must be claimed on the basis of simplicity of rest based approach, it neglects in systematic security analysis and design causing[13], makes it more vulnerable to security risks than the traditional web services. While this is more important to investigate completely from the potential issue of REST based web services before the implementation of system requirements.

CONCLUSION

In this study, we have learned about the application level security to secure the application level security attacks in restful web APIs. The Rest based web services are connected using MVC framework and utilize JSON, XML and plain text as a data exchange format by using ASP.Net Web API. It also leave these web services open to potentially serious attacks

.For example JSON injection attach, cross site scripting attacks etc. used as a thread. In this paper we have investigated the security areas and its techniques used to secure the information of message, which can deliver to the destination safely via expensive protection of method, message encryption by using SSL based security to ensure the confidentiality of data. In chapter 3- Restful security, we tried our best to show the practice to secure the web REST API by using ASP.NET MVC Web API.

REFERENCES

[1] Pistoia, M., &Erlingsson, U. (2008). Programming languages and program analysis for security: A three-year retrospective. *ACM SIGPLAN Notices*, *43*(12), 32-39. doi: 10.1145/1513443.1513449

[2] Jain, A. K., &Shanbhag, D. (2012). Addressing security and privacy risks in mobile applications. IT Professional, 14(5), 28-33. doi: ieeecomputersociety.org/10.1109/MITP. 2012.72

[3] Shar, L. K., & Tan, H. B. K. (2012). Defending against cross-site scripting attacks. Computer, 45(3), 55-62.doi: ieee computer society.org/10.1109/MC.2011.261

[4] Kundu, A., Sural, S., &Majumdar, A. K. (2010). Database intrusion detection using sequence alignment. International Journal of Information Security, 9(3), 179-191.

[5] Werts, J. D., Mikhailova, E. A., Post, C. J., & Sharp, J. L. (2012). An Integrated WebGIS framework for volunteered geographic information and social media in soil and water conservation. Environmental Management, 49, 816–832
REST, http://en.wikipedia.org/wiki/Representational_state_transfer (Web Access: 30th March, 2014)

[6] Model-View-Controller, http://www.w3schools.com/aspnet/mvc_intro.asp (Web Access:30th March,2014)

[7] The Web API pipeline, http://www.dotnetcurry.com/showarticle.aspx?ID=888 (Web Access: 04th April, 2014)

[8] Http Message Handlers, http://www.asp.net/web-api/overview/working-with-http/http-message-handlers (Web Access: 04th April, 2014)

[9] Kennedy, S., Stewart, R., Jacob, P., & Molloy, O. (2011). StoRHm: A protocol adapter for mapping SOAP based web services to RESTful HTTP format. Electronic Commerce Research, 11(3), 245-269. doi: 10.1007/s10660-011-9075-3

[10] Shehab, M., & Marouf, S. (2012). Recommendation models for open authorization. IEEE Transactions on Dependable and Secure Computing, 9(4), 583-596. doi: 10.1109/TDSC. 2012.34

[11] Manago, W. (2011). Protect, maintain information integrity to reduce business risk. *Information Management*, *45*(3), 36-41.

[12] Comerford, C., & Soderling, P. (2010, February 24). Why REST security doesn't exist? *Computerworld*, Retrieved from http://www.computerworld.com/s/article/9161699/Why_ REST_security_doesn_t_exist

[13] Denial of service attachks,http://en.wikipedia.org/wiki/Denial-of-service_attack(Web Access:12th April 2014)

[14] Basic authentication work flow **.** http://www.asp.net/web-api/overview/security/basic-authentication (Web Access:12th April 2014)

[15] Token based security work flow**,** http://leastprivilege.com/page/4/ (Web Access:12th April 2014)

[16] Cross site scripting attacks, http://en.wikipedia.org/wiki/Cross-site_request_forgery (Web Access:12th April 2014)

[17] Cross site scripting attacks, http://www.asp.net/web-api/overview/security/preventing-cross-site-request-forgery-(csrf)-attacks (Web Access: 12th April 2014)

[18] Component of SOA, https://software.intel.com/en-us/articles/modeling-in-the-service-oriented-architecture (Web Access: 12th April 2014)

[19] SOA , http://en.wikipedia.org/wiki/Service-oriented_architecture (Web Access:14th April 2014)

[20] SOAP, http://en.wikipedia.org/wiki/SOAP (Web Access: 14th April 2014)

.