

# Information Prone LDAP Garbage Dumps

Diving in Web Pool

Aditya K Sood aka 0kn0ck. SecNiche Security  
[www.secniche.org](http://www.secniche.org)

---

2007© All Rights Reserved.

This document is a copyright work. SecNiche makes no representation or warranties, either express or implied by or with respect to anything in this document, and shall not be liable for any implied warranties of merchantability or fitness for a particular purpose or for any indirect special or consequential damages. No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any form or by any means, photocopying, recording or otherwise, without prior written consent of SecNiche. While every precaution has been taken in the preparation of this publication, this publication and features described herein are subject to change without notice.

**Foreword:**

I would like to pay my regards to **Mr. Shreeraj Shah**, for reviewing this paper and knowledge sharing on web application security.

Thanks  
Author

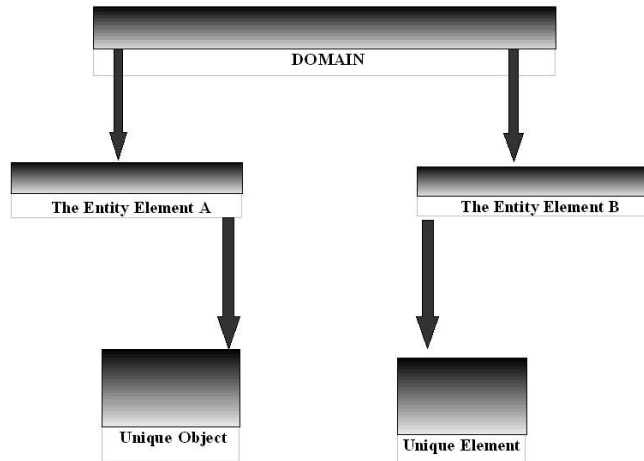
## [1] LDAP Analogy.

*The LDAP garbage dump that remains on web server results in information disclosure. Security of LDAP may be compromised, if for instance a search engine crawls through untamed directories on the web server and finds information through the ldap.xml file. This type of harvesting attack is also termed “static information leveraging attack.” This article provides methods for dealing with this type of attack and clarifying how to secure LDAP.*

The ldap.xml file, often remains on the server due to either misconfiguration or improper server administration. Before I can get to the core, it is essential to understand the anatomy of the LDAP (Lightweight Directory Access Protocol) entities and how these objects work. The protocol is a communication medium. The base of the LDAP protocol is a client server mechanism. LDAP is used to access and update the information in the directory built on the X. 500 mode of communication. This model is used for centralizing data in a specific directory and arranges data in a hierarchical namespace. It further provides powerful search capabilities to track down the required object from the namespace tree. It provides an interface to incompatible directory services. . The great benefit of using LDAP is that it provides a very high speed method for managing directory services over TCP/IP.

The protocol working stats are defined as underlined:

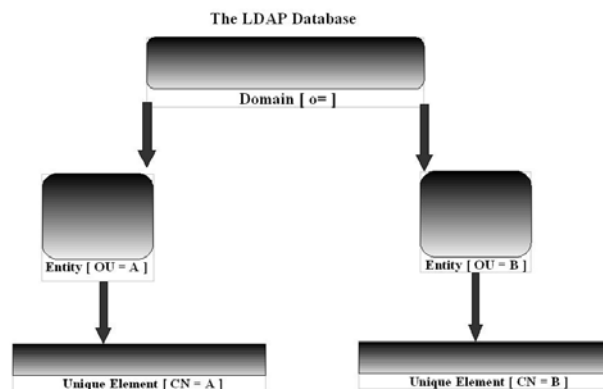
- 1) It possesses a global naming model and unique entries. It means the objects that are created in the centralized directory hold unique properties. It reduces the complexity of interdependency among the objects. In newer versions of LDAP, the naming model is extensible (i.e. schema changes are relatively easy).
- 2) The communication channel is a client server mechanism. In this the specifications for the LDAP are defined. The specifications define the content of messages between the client and server. They are crucial because they define the working model of LDAP itself. The networking stack used is TCP/IP instead of OSI. The OSI is the Open Source Interconnection model of networking having seven layers. This is the prime base of networking but with advent of technology the TCP/IP model is designed. The inbuilt functions used for transactions are simplified. The data is transferred by string generation and delivery model. No specific ASN. 1 notation is used in the LDAP. The ASN relates to Abstract Syntax Notation which encompasses the base encoding rules.
- 3) The LDAP manages clients with well defined agents. The agents are the full designed services that manage the creation and operation of clients who establish connections to the server. It provides extensive security by using Secure Sockets Layer (SSL). LDAP is known for its beneficial application over SSL connections. Moreover the protocol is asynchronous.
- 4) The transferred data s is encoded in a well formed manner, relying on Base Encoding Rules. The encoding depends on the restriction applied to the elements. The element can use bit string or octet encoding for data transaction through the network. The protocol is designed to run over connection oriented and reliable transports. Notice that LDAP does not use Return Referral mechanism i. e. When connection is initiated from clients with referral specified, the protocol returns no referral back to the client. The referrals are a set of URL's that are used by the primary domain server to contact other server to resolve the query in a queue. If the server is inaccessible then multiple URL's are included in the directory object. The protocol provides functionality to the servers to rewrite query for the client. The referral process is done inside the client library and is fully transparent to the applications. This is because all the operations are applied on the centralized directory. In case of any exception, only error structure is replied back to the client. This reduces working complexity of protocol in handling distinguished objects.



Now that I have defined the peripheral layout, let me share how the main design of LDAP protocol is applied. The underlined structure defines the basic category and dissections associated with the predefined structure. This will set you in the LDAP pool and the configurations of object persisting in the specified domain. At the top level, the LDAP centralized database is defined. The database is domain specific, . The domain is always described with notation [O=]. It constitutes the class of entities holding unique elements in them. The elemental entities are defined with [OU=] notation. The object classes are defined to generate a hierarchical tree, referred as Directory Information Tree (DIT), which makes the traversing of directories very efficient. The classes are created as

*[OU=A], [OU=B]*

The classes hold the unique elements, represented in the directory structure as [CN =] The properties defined for all the unique objects are distinguished from each other.. The protocol works on the centralized structure in which the number of classes is designed to integrate the individual objects together. This reduces the distributive effect and need to collect and access objects from different resources. Instead, a pool is created where every operation is performed. The subschema sub entries added to the tree constitute descriptive and unique identifiers. The descriptive Model of LDAP:

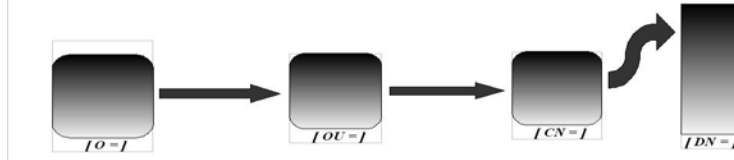


The diagram above shows how the directory consists of the domain, entity element and the unique object, subjugated in the configuration file. This encompasses the architectural behavior of the LDAP protocol. This is a generic layout, but when we want to encounter the whole Domain things are quite different. As the

unique object is created there must be a distinguished name that must be processed. [DN] identifier is used as the global identifier for that unique object

[ DN = ]

The DN is applied as:



That basically encapsulates the fundamental structure of an LDAP. We can now deal with the layout, which is essential since the garbage dump file leverages most of the vital directory information. The next section deals with the structure of the LDAP.xml file

**[2] Design of LDAP.xml File:** It is necessary to understand the core structure of an ldap.xml file before securing your LDAP system. The ldap.xml file is the resource handler configuration file, used for retrieving information from the LDAP user directories. The different properties, preferences and LDAP parameters are configured using the <XML Resource Handler Tag>. Under this tag various specifications are defined which are enumerated below

**[2.1] Structural Property Specification:** The very first specification is the property tag used to configure the properties of the resources. Before analyzing the property tag, it is important to keep in mind that the name of the property in the xml file should be same as the field name of the view. The property specification is divided into four specific objects enumerated as name, type, purpose and display name. These objects set the property of a specific resource.

```
Tag Layout: <properties>
<Propertyname="" type="" purpose="" displayname=""/>
</properties>
```

This is basically how a property specification id is defined.

**[2.2] Structural Preference Specification:** This specification is used to define the preferences of each resource. The preferences play a critical role in ensuring that the file actually sets the parameters for the working stat of that particular object or resource. The preferences also define the context in which the defined resource will be functional. The preference holds secondary objects that define its functional behavior. Let us look at the secondary objects one by one:

- a) The Search scope tag is used to specify the different search values.
- b) The Referral chasing tag for taking external or defined values.
- c) The Timeout tag is used to define the timeout for specific search.
- d) The Authentication User tag to specify the authenticating user.
- e) The Authentication Password to specify the authenticating password.

The authentication credentials are specified within this. This means the working approach of access and authentication mechanism is predefined.

Tag Layout:

```
<Preferences>
<Search Scope></Search Scope>
```

```

<Referral Chasing></Referral Chasing>
<Time Out></Time Out>
<Size Limit></Size Limit>
<Authentication User></Authentication User>
<Authentication Password></Authentication Password>
</Preferences>

```

**[2.3] Structural Specification For LDAP Server Path:** The LDAP server requires a particular path for every single object or resource designed in the domain context (i.e. to make the resource functional). in order to access it. Tag Layout :

```
<LDAPServer>LDAP://yourldapserver:389/</LDAPServer>
```

This predefined specification looks similar to an HTTP URL and thus provides an easy method for configuring the path for LDAP server.

**[2.4] Structural Specification for LDAP Search Base:** This specification holds the context of the subschema entries or their sub entries. This tag is used to design the directory component. The directory holds the record of every single unique object. The component creation defines the search base for different classes. Tag Layout:

```
<LDAPSearchBase>dc=yourdc,dc=com</LDAPSearchBase>
```

The search base is really crucial for any LDAP operation to occur because in absence of search base the traversing of tree becomes complex or impossible.

**[2.5] Structural Specification For LDAP Base Filters:** This specification is important since the tag is used to design search filters for the directory traversal. The filters play a vital role in search anatomy. The proper design of the filter makes work easier and reduces complexity. Tag Layout:

```
<LDAPBaseFilter>(objectCategory=Person)</LDAPBaseFilter>
```

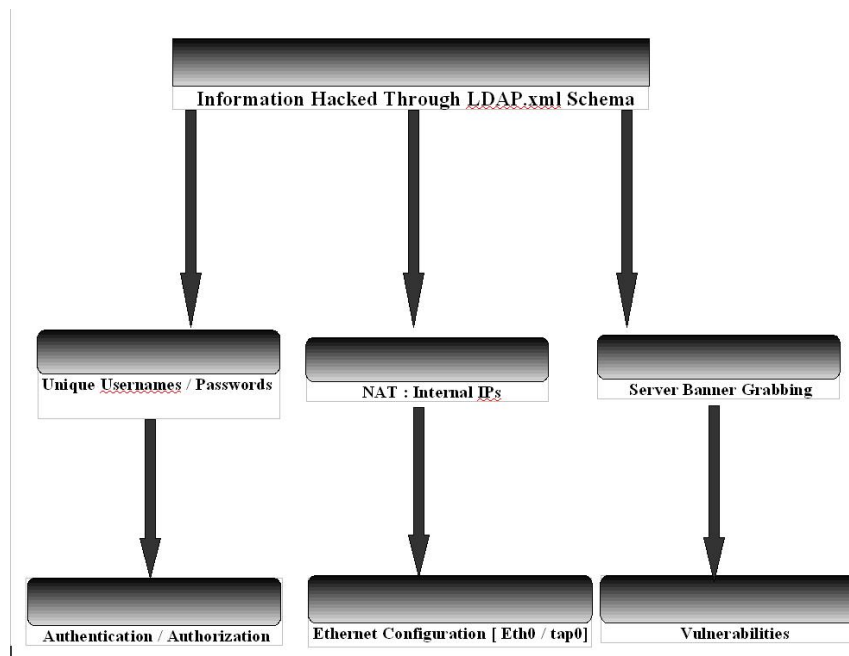
The filters are created uniquely, which means one can design individual filters for each unique user. The filters are applicable only to those objects for which these are predefined. Let's look at a simple ldap.xml file:

```

<XmlResourceHandler>
<Properties>
<Property name="user" type="string" displayName="User"/>
<Property name="distinguished Name" type="string" purpose="identifier"
displayname="Distinguished Name"/>
<Property name="name" type="string" purpose="name" displayName="Name"/>
<Property name="mail" type="string" purpose="email" displayName="Email"/>
</Properties>
<Preferences>
<Search Scope>Sub Tree</Search Scope>
<Referral Chasing>External</Referral Chasing>
<Time Out>60</Time Out>
<Authentication User></Authentication User>
<Authentication Password></Authentication Password>
<LDAPServer>LDAP://yourldapserver:389/</LDAPServer>
<LDAPSearchBase>dc=yourdc,dc=com</LDAPSearchBase>
<LDAPBaseFilter>(objectCategory=Person)</LDAPBaseFilter>
</XmlResourceHandler>

```

**[3] Ldap.xml Information Model:** This model is generic and provides the kind of information that gets extracted from the garbage dump of the LDAP.xml file. The model focuses on the key information that can be revealed from the static xml file. The static file holds specification for a particular service and how configurations are made. Before getting to garbage dump concept one must understand the model. The model describes the different types of information based on the different classes of attacks. The model in whole is fully inadvertent to the ldap.xml file. Often, the garbage dumps are the result of poor web administration or mis configuration by the developers or service providers. It is very important to check the data directories for untamed information for keeping the server integrity intact. The information extracted gives us an outline of the services and configuration of the objects on the server.



The diagram reflects the classes and types of information leveraged out of the files. It depends on the the configuration and preferences set for the resource placed in the server.

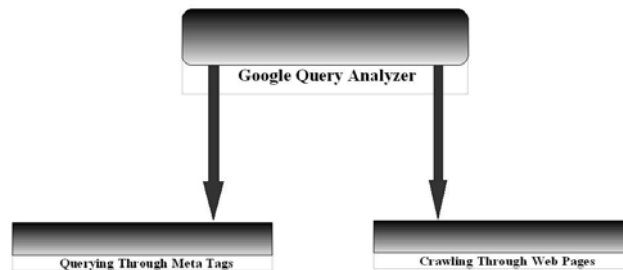
**[3.1] Authentication And Authorization:** The authentication credentials are defined in the ldap.xml file in a very static manner. Thus, user names and passwords once entered tend to be highly static. This means that if someone steals the ldap.xml file, the entire authentication scheme can potentially be identified. This also depends on the configuration parameter of the server in which this file has been used. The possibility is that someone may steal the present credentials from the ldap.xml file. Moreover this is a relatively flat model of security in which the transaction is accompanied without randomizations, which means the parameter used in the configuration is not generated in a random manner. This vector gets deeper because once you have the credentials you can access the other services defined for that particular group of users. Next is the authorization mechanism. As an attacker, taking control over the authentication scheme, the authorization scheme can then be manipulated for leveraging further information. Therefore, if proper access control is not applied to the files, the security implications are severe. The security of the xml file is of prime importance.

**[3.2] Internal Information:** The ldap.xml file discloses a lot of internal information regarding the Network Address Translation (NAT) to set the mapping from external objects to internal ones. The (NAT) is the internal mapping of system IP addresses. . Therefore, attackers can potentially determine the kind of scheme used in NAT. The internal IP addresses may also get harvested. Moreover, even the number of network Ethernet interfaces can be identified from the LDAP file. The disclosure of internal information

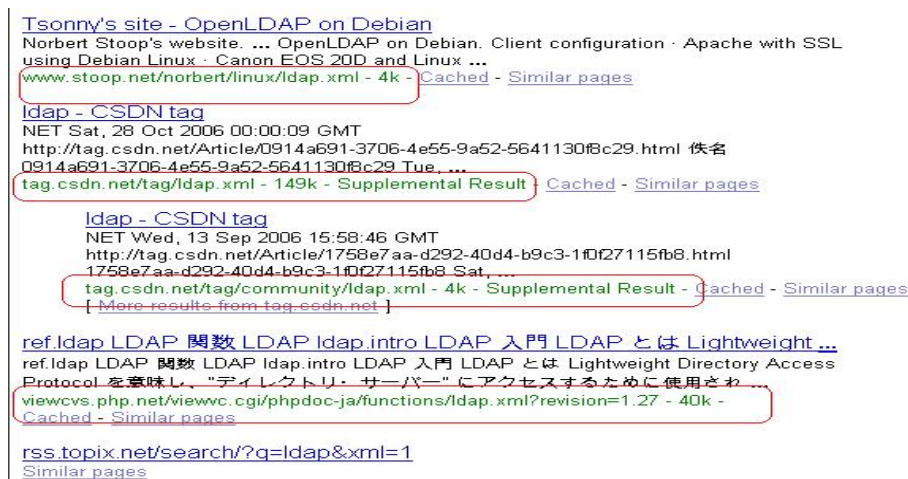
works in a chain reaction. This is one negative aspect of insecure xml files of LDAP. The negative aspect covers the unhandled security information in the xml file. The information relates to system configuration parameters, request-response handling, error status codes etc. Every single object is provided with access parameters. If these parameters are not defined well results in internal information disclosure.

**[3.3] Peripheral Information:** The peripheral information like header grabbing is also possible, which means someone can harvest the information regarding the version and kind of running servers and applications . This becomes as the attack base because an attacker searches for the required vulnerabilities from the security databases. If any potential vulnerability is found the system security may be compromised. It depends on whether the parameters are applied with security and with obscurity

**[4] Stroking Google to Find Garbage Dumps:** It is now time to start penetrating into your network to find weak links that provide information regarding the LDAP garbage dumps. Our trusty friend and tool name Google can help with this. Lets first look into how the Google searches for the file.



Google uses strategically different techniques based on the queries defined in the search path. The very first technique is based on **MetaTag Traversing** (basically crawling through the meta tags of web pages). The second technique is to do text specific **Web Page Crawling**. Note that this includes traversing your web page code for comments and relative names. This makes it easy to ping the servers for the desired results. The LDAP.xml garbage files are often far too easy to find. For example, look over the screen-shot below: Google search show lots of ldap.xml files, which are often integrated together in a search page. This is a serious security weak point. Although not all files may be useful, a number of files do hold importance from a security perspective. The link to the files possesses information regarding the network of the specific organization. That means that there is already the exposed vulnerability of **Dumpster Diving in Web Pool** because a lot of information can be extracted via the server with a nominal effort. This technique rather I say is an art and is very crucial. An attacker launches a simple query like:



**inurl:ldap.xml site:com**  
**allinurl:indexof ldap.xml site:org**

The defined queries are constructed by using search elements of Google. LDAP garbage dumps therefore present a serious problem if they expose network and DIT to the web. The DIT stands for Directory Information Tree. It encapsulates the hierarchical information in the form of tree pattern that makes the traversing of objects easier.

**[5] Cause of Garbage Dumps:** As we know the xml file is processed by the service that is running on the server. Whenever an application is invoked, the service looks into the ldap.xml file to see the configurations. The security flaw occurs in the application of deployment descriptors that provides security to the files such as:

```
<web-service>
  name=service_name
  uri=service_uri
  exposeXml="True"
  exposeHomePage="False"
</web-service>
```

The configuration parameters for the xml file are specified in the web service tag. The garbage dumps are the result of poor administration. The exposed XML is not properly configured in a file on the web servers [web.xml] that let the attackers crawl the xml files on the using the web server as an entry point. The server containers hold the working of the LDAP xml files. It is therefore important to lock the files (example: chmod 700) or, place files in a location, not available to the web server (out of the httpd path). If a search bot crawls and finds such a link to the LDAP.xml files and the link is displayed on the search engine an attacker can get to the links and parse them to see the xml schema file for the LDAP protocol. It is therefore very important to focus security efforts around the ldap.xml file because it consists of centralized information such as users of the whole network of any organization.

**[5.1] Case Study:** The case study is an outcome of penetration session that I undertook to prove this concept. I will provide you with an example in which you will encounter about information disposed off by ldap.xml files. This example is very clear and context driven. The type of information leveraged will be the analyzing aspect of web penetration. The process I followed is same as described above. We are going to traverse ldap.xml file and analyze the bulk of information extracted. A tiny mistake in configuration leads to untamed consequences.

**The Example:** With the help of this layout I will prove the desired concept which we have already undertaken. The very first element is to traverse the web for having ldap based xml files. The point is , to parse every single link to find the target garbage dump file which automatically reveal the information. Always remember 90% information extracted is fully functional and is driven with originality.

```
- <dia:object type="Cisco - WWW server" version="0" id="013">
  - <dia:attribute name="obj_pos">
    <dia:point val="3.975,7.725"/>
  </dia:attribute>
  - <dia:attribute name="obj_bb">
    <dia:rectangle val="3.947,6.725;6.27053,10.258"/>
  </dia:attribute>
```

You can enumerate, the target is using "Cisco Server" with version specification and id. It provides lot of information which can be enlarged by searching databases on the internet. The version and id makes the attacker to search for potential vulnerabilities for this Cisco server. Afterwards only object attributes are defined. This inherits the information of a Cisco router.

```

- <dia:composite type="text">
  - <dia:attribute name="string">
    <dia:string>#192.168.77.1 / tap0#</dia:string>
  </dia:attribute>

```

In the above snapshot target is using Network Address Translation. The internal network is configured with class C address. The interface used in this is tap0. A crystal clear picture of the target interface configuration is provided.

```

- <dia:attribute name="string">
  <dia:string>#192.168.77.2 / eth0#</dia:string>
</dia:attribute>

```

This layout provides us information about network which is configured with different interfaces and multiple IP addresses. At this point we are going to have look at the HTML output of the file. This will help us to analyze the information that is present on the web.

---

```

#A4# #Sperimentazione di server LDAP con replica# #Id: ldap.xml,v 1.2 2005/04/08 14:12:26 doros Exp $# #Virtual space# #192.168.50.2 / eth1# #Real
space# #192.168.77.1 / tap0# #192.168.77.2 / eth0# #RealHost# #A# #192.168.50.1 / eth0# #192.168.50.3 / eth0# #rserver.istituto.it# #server.istituto.it#
#client.istituto.it# #192.168.50.0/29# #(ldap.istituto.it & ns.istituto.it)# #(ldap2.istituto.it)#

```

The information is almost the same that we have analyzed. So information leakage occurs with minimum intervention in the network. This provides a subtle base for the attackers. When attacker possesses information of a particular network then exploitation can occur. An error generated at the top level will go on augmenting when traverses down the hierarchy. It is stated as Top to Bottom Error Chaining. A simple configuration mistake results in the generation of attack base. The another example I am going to present is from Freshmeat portal. This is general example but the information can be undertaken regarding Uniform Resource Locators and the parameters used in that.

```

45515 2004-03-15 04:48:04 2004-03-17 05:51:56 checkpassword-ldap Checkpassword-LDAP An implementation of the checkpassw
authentication program. Checkpassword-LDAP is an implementation of an LDAP authentication program that is compatible with checkpa
&quot;checkpassword&quot; interface are available at http://cr.yo.to/checkpwd.html. 1.00 0.00 22885 159.42 0.28 17099 0.00 0 344 ;
http://freshmeat.net/projects/checkpassword-ldap/ http://freshmeat.net/redirect/checkpassword-ldap/48658/url_homepage/
http://freshmeat.net/redirect/checkpassword-ldap/48658/url_bz2/ GNU General Public License (GPL) 0.01 154383 2004-03-15 07:42:44
http://freshmeat.net/~rvcommowick/ Owner The Treuf http://freshmeat.net/~Treuf/ Admin 9 15 201 164

```

This concept is based on the fact that with minimum intervention in the web pool you can extract lot of information. This is crucial because everything is on the fly. After getting to this you will also think about LDAP injection scenario. That's a good part but the injections are checked against dynamicity of the web application but this lays stress on information prone garbage files that remain there in the web directories. This concept inherits staticity rather dynamicity. A garbage dump ldap.xml file present on the web server will act as the base of exploitation for that generic server. The human malfeasance factor is always a weak spot in application and system security.

**[5.2] Configuration Parameter Checks:** The configuration parameter check is very crucial for defining the working state of a particular service based on the definite xml file. The configuration arguments should be strictly defined against the objects inherited in the server. The major cause of information leaks is wrong configuration of files on the server, making it vulnerable to manipulation. For example configuration in the xml file can mitigate the risk to some extent by not allowing the exposure of xml files:

```
<web-service>
name=service_name
uri=service_uri
exposeXml="False"
exposeHomePage="False"
</web-service>
```

The above stated example is restricting the web server's exposure of the xml file to the public. This sets a basic level of security. The configurations should be made with restrictions in mind.

**[5.3] Meta Tag Element Restriction:** With the rise of search engines as tools for hacking, the Meta tag must also be taken into consideration as a security factor. One should adopt Passive security methodology. This means one should not add private keywords to the application web page. The meta tags are defined as :

```
<META HTTP-EQUIV="content-type" CONTENT="text/html; charset=iso-8859-1">
<meta name="keywords" content="LDAP XML etc">
```

The above defined Meta tags are used to search the content of LDAP, XML etc. This usually means the web crawler tagged the link of this web page against the search for LDAP and XML. Never expose any facet of how your back-end operates.

**[5.4] Web Administration Checks:** A substantive number of attacks flourish through the network because of poor system administration. It becomes imperative to check the service stats of the server regularly. An ingress testing for the xml configuration files with local parameters should be performed continuously. To prevent security vulnerabilities, any unnecessary web pages or xml files should be removed by the web administrators.

## **[6] Conclusion:**

LDAP offers administrators a powerful tool for managing directory services. However, with this power, comes the responsibility to ensure that security holes are tightly sealed. One of the major vulnerabilities is the LDAP garbage dump file. This can be reconciled by understanding how LDAP uses the file. There is also a need to appreciate the importance of securing the LDAP.xml file and administering the web server properly. One minor mistake can open a major security hole. Therefore, several levels of security should be applied around any LDAP use. Finally, any security model must also take into account the human factor in order to have a robust and safe LDAP implementation