Network Working Group
Request for Comments: 3663
Category: Experimental

A. Newton
VeriSign, Inc.

December 2003

Domain Administrative Data in Lightweight Directory Access Protocol (LDAP)

Status of this Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Copyright Notice

Abstract

Domain registration data has typically been exposed to the general public via Nicname/Whois for administrative purposes. This document describes the Referral Lightweight Directory Access Protocol (LDAP) Service, an experimental service using LDAP and well-known LDAP types to make domain administrative data available.

Table of Contents

1. Introduction

This document describes the Referral Lightweight Directory Access Protocol (LDAP) Service, an experimental project launched by VeriSign, Inc., to explore the use of LDAP and LDAP-related technologies for use as a directory service of administrative domain registration information.

1.1. Historical Directory Services for Domain Registration Data

The original National Science Foundation contract for the InterNIC called for the creation of an X.500 directory service for the administrative needs of the domain registration data and information. Due to problems with implementations of X.500 server software, a server based on the Nicname/Whois [1] protocol was temporarily erected.

In 1994, the Rwhois [3] protocol was introduced to enhance the Nicname/Whois protocol. This directory service never gained wide acceptance for use with domain data.

Presently, ICANN requires the operation of Nicname/Whois servers by registries and registrars of generic Top-Level Domains (TLD's).

1.2. Motivations

With the recent split in functional responsibilities between registries and registrars, the constant misuse and data-mining of domain registration data, and the difficulties with machine-readability of Nicname/Whois output, the creation of the Referral LDAP Service had the following motivations:

- Use a mechanism native to the directory protocol to refer clients from inquiries about specific domains made at a registry to the appropriate domain within the appropriate directory service at a registrar.
- Limit access to domain data based on authentication of the client.
- Provide structured queries and well-known and structured results.
- Use a directory service technology already in general use.

Given these general criteria, LDAP [5] was selected as the protocol for this directory service. The decision was also made to restrict the use of LDAP to features most readily available in common implementations. Therefore, a goal was set to not define any new object classes, syntaxes, or matching rules.

The experiment was successful in exploring how LDAP might be used in this context and demonstrating the level of customization required for an operational service. Conclusions and observations about this experiment are outlined in Section 6.

1.3. Abbreviations Used

The following abbreviations are used to describe the nature of this experiment:

TLD: Top-Level Domain. Refers to the domain names just beneath the root in the Domain Name System. This experiment used the TLD's .com, .net, .org, and .edu.

SLD: Second-Level Domain. Refers to the domain names just beneath a TLD in the Domain Name System. An example of such a domain name would be "example.com".

DIT: Directory Information Tree. One of many hierarchies of data entries in an LDAP server.

DN: Distinguished Name. The unique name of an entry in a DIT.

cn: common name. See RFC 2256 [7].

dc: domain component. See RFC 2247 [4].

uid: user id. See RFC 2798 [9].

2. Service Description

The service is composed of three distinct server types: a registry LDAP server, registrar LDAP servers, and registrant LDAP servers.

The registry LDAP server contains three Directory Information Trees (DIT's).

- The Top-Level Domain DIT's follow the DNS hierarchy for domains (e.g., dc=foo,dc=com).
- The name server DIT allows a view of the name servers, many of which serve multiple domains.
- The registrar-referral DIT provides referrals from the registry into the respective TLD DIT of the registrars (on a TLD basis).

The registrar LDAP server contains two types of DIT's.

- The TLD DIT follows the DNS hierarchy for domains (e.g., dc=foo,dc=com) and parallels the TLD DIT of the registry.

- The name server and contact DIT allow a view of the name servers and contacts, many of which are associated and serve multiple domains.

There is no specification on the DIT or schema for the registrant LDAP server. Referrals from the registrar server to the registrant server are provided solely for the purpose of allowing the registrant direct control over extra administrative information as it relates to a particular domain.

Access control for this service is merely a demonstration of using a Distinguished Name (DN) and password. Should registries and registrars uniformly adopt LDAP as a means to disseminate domain registration data, standardization of these DN's would need to be undertaken based on each type of user base.

3. Registry LDAP Service

3.1. TLD DIT

3.1.1. DIT Structure

The registry TLD DIT has the following structural hierarchy:

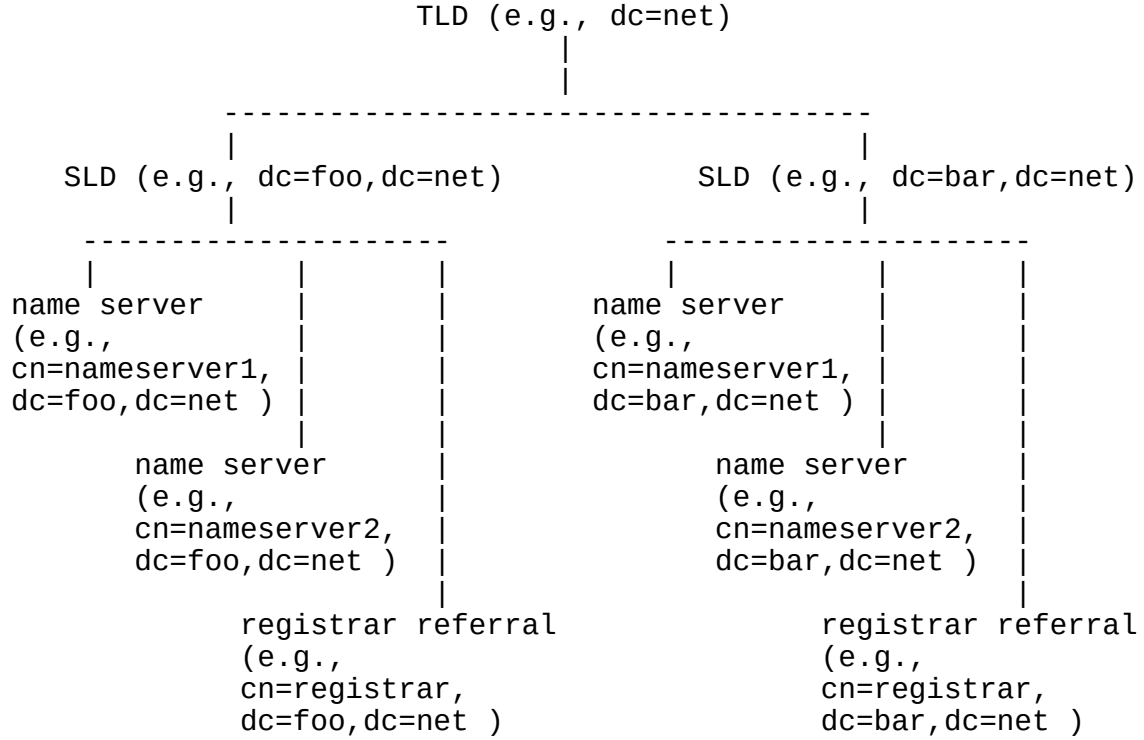

Figure 1: Registry DIT Overview

The root of a TLD DIT is an entry of objectclass domain as specified by RFC 2247 [4] and represents a top-level domain.

The second tier of the DIT represents second-level domains. Each of these entries is of objectclass domain as specified by RFC 2247 [4]. The description attribute on these entries often contains descriptive text giving the name of the registrar through which these domains have been registered.

The third tier contains entries specific to each second-level domain. Name server entries are of objectclass ipHost as specified by RFC 2307 [8]. The distinguished names of these name server entries are algorithmically calculated, where the first component is the word

"nameserver" concatenated with an index number of the name server entry and the remaining components are the appropriate domain names. There is no specification relating the value of the name server entry to the index it may be assigned other than it is unique and consistent with respect to the client session. This tier also contains the referral from the registry to the registrar. This referral is a direct referral to the entry in the appropriate registrar LDAP server corresponding to the domain name that the referral falls beneath in this DIT.

3.1.2. Allowed Searches

Because of the vast number of entries contained within this DIT, only certain types of searches are allowed. Allowing any search expressible via LDAP would lead to expensive searches that would be far too costly for a publicly available service. The searches allowed are as follows:

- One-level scoped searches based at the root of the DIT. Substring matching is allowed on dc attributes, but the substring must be at least be 3 characters in length.

- Base search based at the root of the DIT.

- Base, one-level, and sub-tree searches based at any second level domain name (the second tier) and below.

3.1.3. Access Control

The registry TLD DIT only has one access control type. When a client binds with a DN of "cn=trademark" and password of "attorney", the second-level domain entries also take on an objectclass of extensibleObject with the added attributes of "createddate" and "registrationexpirationdate", which are of type Generalized Time, as specified by RFC 2252 [6].

3.2. Name Server DIT

3.2.1. DIT Structure

The registry name server DIT has the following structural hierarchy:

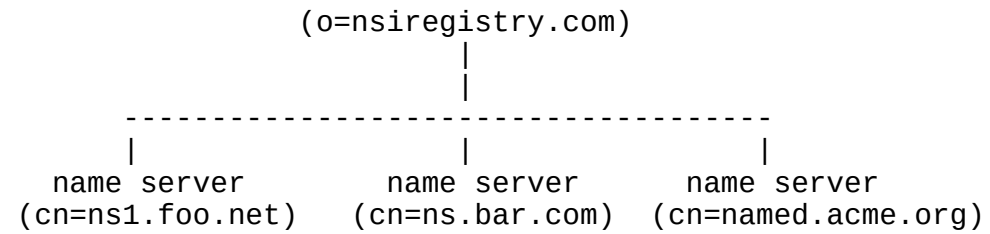

Figure 2: Registry DIT Overview

The root of a name server DIT is an entry of objectclass organization as specified by RFC 1617 [2]. It has no significance other than to serve as the root of the DIT.

The second tier of this DIT represents name servers. Each of these entries is of objectclass ipHost, as specified by RFC 2307 [8].

3.2.2. Allowed Searches

Because of the vast number of entries contained within this DIT, only certain types of searches are allowed. Allowing any search expressible via LDAP would lead to searches far too costly for a publicly available service. The searches allowed are as follows:

- One-level and sub-tree scoped searches based at the root of the DIT if a filter on the cn attribute is provided.
- Base search based at the root of the DIT.
- Base, one-level, and sub-tree searches based at any name server entry.

3.3. Registrar Referral DIT

3.3.1. DIT Structure

The registry registrar-referral DIT has the following structural hierarchy:

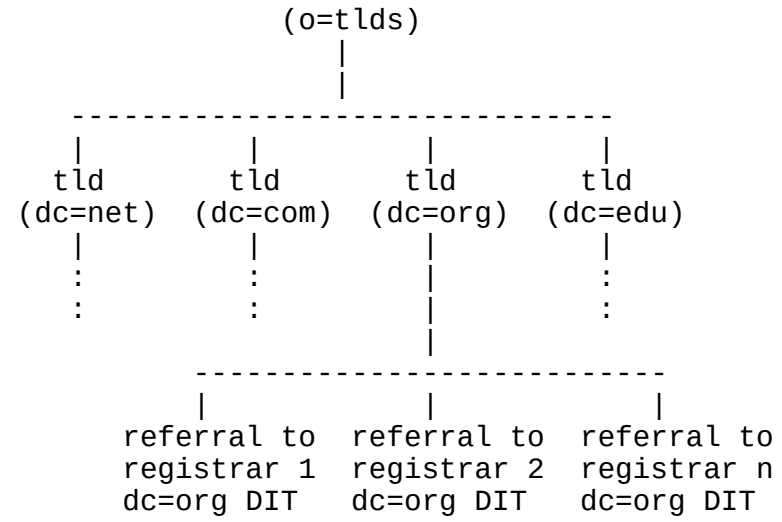

Figure 3: Registry Referral DIT Overview

The root of the registrar referral DIT is an entry of objectclass organization, as specified by RFC 1617 [2]. It has no significance other than to serve as the root of this DIT.

The second tier of this DIT represents top-level domains. Each of these entries is of objectclass domain, as specified by RFC 2247 [4].

Underneath each TLD entry, the third tier contains referrals to the appropriate TLD DIT of each registrar.

4. Registrar LDAP Service

4.1. TLD DIT

4.1.1. DIT Structure

The registrar TLD DIT, which is similar to the registry TLD DIT, has the following structural hierarchy:

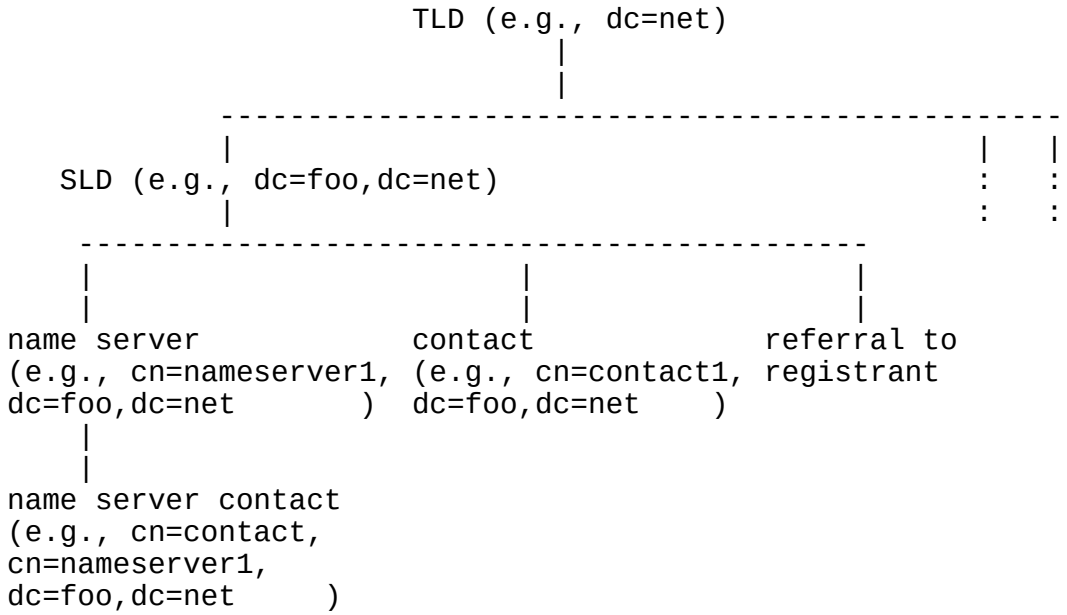

Figure 4: Registrar DIT Overview

The root of a TLD DIT is an entry of objectclass domain, as specified by RFC 2247 [4] and represents a top-level domain.

The second tier of the DIT represents second-level domains. Each of these entries is of objectclass domain, as specified by RFC 2247 [4].

The third tier contains entries specific to each second-level domain. The entries at this level are as follows:

- Name server entries are of objectclass ipHost, as specified by RFC 2307 [8]. The distinguished names of these name server entries are algorithmically calculated where the first component is the word "nameserver" concatenated with an index number of the name server entry and the remaining components are the appropriate domain names. There is no specification relating the value of the name server entry to the index it may be assigned other than it is unique and consistent with respect to the client session.

- Contact entries are of objectclass inetOrgPerson, as specified by RFC 2798 [9]. The distinguished names of these contact entries are algorithmically calculated, where the first component is the word "contact" concatenated with an index number of the contact and the remaining components are the appropriate domain names. There is no specification relating the value of the contact entry to the index it may be assigned other than it is unique and consistent with respect to the client session. The description attribute of the entry contains the role for which a contact is related to a domain. These roles are identified as "Admin Contact", "Technical Contact", and "Billing Contact", and may appear in any order.

- Finally, this third tier contains the referral from the registrar to the registrant.

The fourth tier only contains name server contact entries. These entries are of objectclass inetOrgPerson, as specified by RFC 2798 [9].

4.1.2. Allowed Searches

Because of the vast number of entries contained within this DIT, only certain types of searches are allowed. Allowing any search expressible via LDAP would lead to searches far too costly for a publicly available service. The searches allowed are as follows:

- One-level scoped searches based at the root of the DIT. Substring matching is allowed on dc and o attributes, but the substring must be at least 3 characters in length.

- Base search based at the root of the DIT.

- Base, one-level, and sub-tree searches based at any second level domain name (the second tier) and below.

4.1.3. Access Control

The registrar TLD DIT has two access control types. When binding anonymously, a client only sees dc, o, and c attributes of the second-level domain entries. When a client binds with a DN of "cn=trademark" and password of "attorney", all of the other attributes normally available on entries of objectclass domain are visible if they have values. In addition, if a client binds with the DN of a contact and password of "password", all attributes for second-level domain entries for which the bind DN has a relation are visible.

4.2. Name Server and Contact DIT

4.2.1. DIT Structure

The registrar name server and contact DIT has the following structural hierarchy:

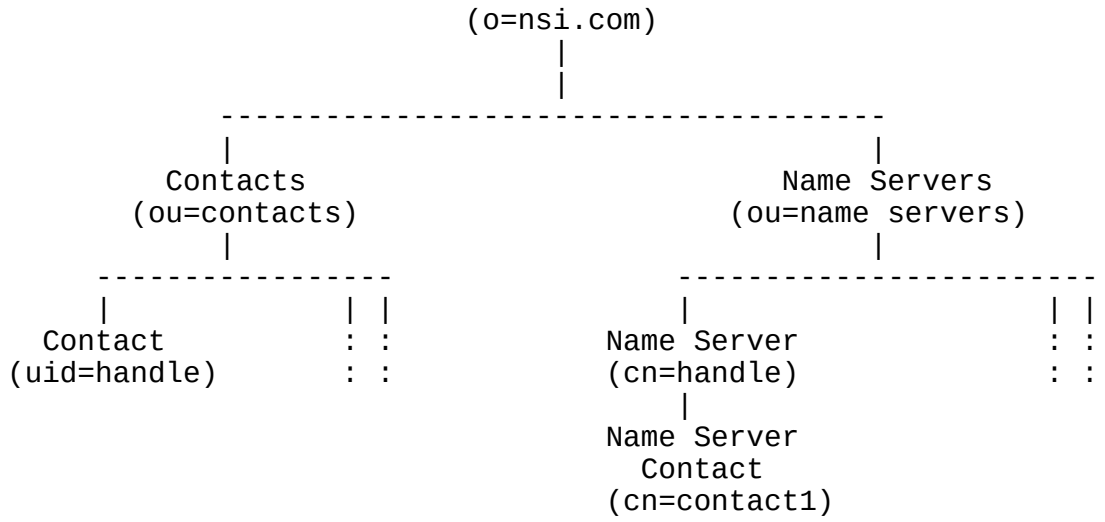

Figure 5: Registrar DIT Overview

The first tier of the name server and contact DIT is an entry of objectclass organization, as specified by RFC 1617 [2].

The second tier of the DIT contains two entries, each of which is of objectclass organizationalUnit, as specified by RFC 2256 [7]. One entry represents the part of the DIT containing contacts and the other entry represents the part of the DIT containing name servers.

Entries underneath the contacts organizationalUnit entry are of objectclass inetOrgPerson and represent contacts registered with the registrar. Their RDN is composed of the uid attribute. The uid attribute's value is a unique identifier or handle that is registrar assigned.

Entries underneath the name server organizationalUnit entry are of objectclass ipHost and represent name servers registered with the registrar. Their RDN is composed of the cn attribute. The cn attribute's value is a unique identifier or handle that is registrar assigned. Each name server entry may optionally have children entries of objectclass inetOrgPerson. These entries represent the contacts of the name server they fall beneath.

4.2.2. Allowed Searches

Because of the vast number of entries contained within this DIT, only certain types of searches are allowed. Allowing any search expressible via LDAP would lead to searches far too costly for a publicly available service. The searches allowed are as follows:

- One-level and base searches at the root of the DIT.
- Sub-tree searches at the root of the DIT using cn and uid attributes as a filter.
- Base searches at either entry of the second tier.
- One-level and sub-tree searches at either entry of the second tier, using cn or uid attributes as a filter.
- Base, one-level, and sub-tree searches based at any contact or name server entry and below.

5. Clients

Early scoping and analysis of this project were based on the use of output from command line clients, specifically the "ldapsearch" command present with many implementations of LDAP servers. Our survey of this tool, available from many vendors, showed that referral chasing was difficult to control or predict, and the behavior between these implementations with respect to referral chasing was inconsistent.

Based on the limited nature of the expressive capabilities present with just command line tools, searches involving nested queries or advanced referral chasing were deemed the domain of clients making direct use of LDAP client libraries. Three of these types of clients were produced: a web-based client, a cross-platform C-based client, and a Java client. No significant deficiencies or problems were found with the LDAP client libraries in the construction of these clients, and the level of control provided by their programming interfaces was adequate to create the necessary searches. Instead, most of the problems encountered with these clients were based on usability concerns.

It was found that the web-based client caused a great amount of confusion for users not familiar with LDAP or Nicname/Whois with respect to the underlying technology and the network model. Thus, many users believed the web-based client to be the only interface to the data and were unaware or confused by the intermediate LDAP protocol. In addition, it was difficult to express to users the

registry-registrar-registrant service model in adequate terms from search results where the results could be rendered properly among the various common web browsers.

Both the C and Java based clients were built to be both graphical and cross-platform (in the case of the C-based client, the Linux and Windows platforms were chosen as targets). The LDAP client libraries chosen for both clients proved to be quite capable and offered the necessary levels of control for conducting nested queries and advanced referral chasing. Expectations at the outset for construction of both clients, based on past experience, were that the C-based client would not only perform better than the Java client but also have a better appearance. In reality, these assumptions were incorrect as there was no perceivable difference in performance and the look of the Java client was often considered to be far superior to its counter-part. In addition, the Java client required much less time to create. Both clients are available under the terms of an open source license. Though it is impossible to have accurate measurements of their popularity, through monitoring and feedback it was perceived that the web-based client had far greater use.

6. Lessons Learned

Based on the experience of piloting this experimental service, feedback from users of the service, and general comments and observations of current and common opinions, the following items have been noted.

6.1. Intra-Server Referrals

Original analysis of the data set to be used revealed a high degree of relationships between name servers, contacts, and domains. Storing the data in non-normalized form according to the DIT outlined in this document would make an original relational dataset of roughly 20 million objects explode to over 115 million objects.

To combat this problem, the first pass at defining the DIT's made heavy use of referrals between the TLD DIT's and the name server and contact DIT's. The use of the 'alias' objectclass was considered but ruled out in hopes of using referrals for load balancing across servers (i.e., placing each TLD DIT on a separate server, and separate servers for the name server and contact DIT's). However, initial testing with the 'ldapsearch' command found inconsistencies with the interpretation of the referrals and how they were managed. Not only were the results inconsistent between implementations, but many of these clients would easily get caught in referral loops.

The final solution to the problem was to create a customized back-end data store containing the data in a normalized form. This gave the client the appearance of having a non-normalized data set which required no intra-server referrals. Aliases may have been a better solution, however our interpretation of their output with implementations of the 'ldapsearch' tool was not satisfactory. It was also later learned that some LDAP server implementations place certain restrictions on aliases that would have conflicted with our overall DIT structure. In the end, it was felt that a customized back-end would be required by any server with a large data-set, but smaller data-sets for less populated domains could easily use off-the-shelf implementations.

6.2. Inter-Server Referrals

The modeling of the overall service to provide the split in operational responsibility between registry and registrar required the use of referrals (i.e., the two servers would not be operated by the same organization, therefore would most likely not co-exist on the same physical machine or network). The chief problem with LDAP referrals returned for this purpose grew out of the need to limit data returned to the client and the priority given to referrals. It was quite easy to cause a sub-tree query at certain levels, for instance a TLD level, to return nothing but referrals. This was true because referrals would be returned out of the scope of the supplied search filter and therefore would fill the result set to its limit, normally set to 50 entries.

In certain use cases, a result set with nothing but referrals was desired (e.g., o=tlds). However, even in these cases it was possible for some referrals to not be returned due to the size limit. In this case, it was felt that a result set of 50 referrals, the default for the size limit in most cases, was too large for any practical use by a client and was a failing of query distribution in general rather than a limitation of LDAP.

6.3. Common DIT

Because of the nature of software development, the graphical and web clients were developed after the development of the server software. The 'ldapsearch' client was used for testing and development during server software creation. It was not until the creation of more advanced clients that it was discovered that the design decision of uniform DIT naming should have been made. Technically, this would have allowed for slightly better software modularization and re-use. In addition, the use of a company name in the DIT structure did not allow the easy integration of another domain registry, as in the registry-registrar model. Not only would clients have to be

reconfigured for each new registry operator, but this would most likely have social implications as well.

6.4. Universal Client

The construction of the clients revealed yet another misconception. Though this project used a generic directory service technology, the clients required a high-degree of algorithmic knowledge about the DIT structure and schemas being used. The graphical clients could not be used against an LDAP service with another DIT or schema. Therefore, a generic or universal client, one that could be used for all LDAP applications, would either not be able to make full use of the data provided by the service or would be far too complex for operation by the average user.

6.5. Targeting Searches by Tier

The network model for this service was divided into three tiers: registry, registrar, and registrant. Despite this, all searches needed to start at the registry level causing overhead for searches that could be targeted at a select tier. This service did not implement a solution to this problem, such as using SRV and/or NAPTR records in DNS to allow a client to find a responsible LDAP server.

6.6. Data Mining

Section 3.1.2 and Section 4.1.2 describe the searches allowed by this service. However, the most common question asked by users of the service revolved around getting around these restrictions. Because browsing at the TLD level was not permitted, many users asked about the feasibility of using recursive dictionary queries to circumvent the search restrictions.

It should be noted that many operators of Nicname/Whois server consider this practice to be data mining and often refer to it specifically as a dictionary attack.

7. IANA Considerations

There are no applicable IANA considerations presented in this document.

8. Internationalization Considerations

The domain administrative data in this service did not cover Internationalized Domain Names (IDN’s).

9. Security Considerations

This experiment did not endeavor to use security mechanisms beyond those readily available in LDAP [5]. Section 3.1.3 and Section 4.1.3 describe the various access controls used within the scope of the defined security mechanisms. While these mechanisms were adequate for this experimental deployment, they would not be adequate for a production environment, and they should not be taken as a model for those contemplating deployment on the Internet.

10. Intellectual Property Statement

11. Normative References

[1] Harrenstien, K., Stahl, M. and E. Feinler, "NICNAME/WHOIS", RFC 954, October 1985.

[2] Barker, P., Kille, S. and T. Lenggenhager, "Naming and Structuring Guidelines for X.500 Directory Pilots", RFC 1617, May 1994.

[3] Williamson, S., Kosters, M., Blacka, D., Singh, J. and K. Zeilstra, "Referral Whois (RWhois) Protocol V1.5", RFC 2167, June 1997.

[4] Kille, S., Wahl, M., Grimstad, A., Huber, R. and S. Sataluri, "Using Domains in LDAP/X.500 Distinguished Names", RFC 2247, January 1998.

[5] Wahl, M., Howes, T. and S. Kille, "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997.

[6] Wahl, M., Coulbeck, A., Howes, T. and S. Kille, "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions", RFC 2252, December 1997.

[7] Wahl, M., "A Summary of the X.500(96) User Schema for use with LDAPv3", RFC 2256, December 1997.

[8] Howard, L., "An Approach for Using LDAP as a Network Information Service", RFC 2307, March 1998.

[9] Smith, M., "Definition of the inetOrgPerson LDAP Object Class", RFC 2798, April 2000.

Appendix A. Other Work

In addition to the deployment of servers and development of clients, VeriSign conducted two sub-projects related to this experiment.

The first project was a Nicname/Whois-to-LDAP gateway. The goal of the project was to create an LDAP server for use by registrars to deploy in front of their Nicname/Whois servers. This gateway would take LDAP requests, translate them to Nicname/Whois requests, issue the request to a specific Nicname/Whois server deployed on port 43, interpret the response, and return LDAP result sets. Because of the unspecified nature of Nicname/Whois result sets, the gateway was programmed to specifically recognize only the output of three distinct registrars. While this gateway proved valuable enough to allow domain lookups and limited searches, it was unable to provide consistent contact lookups, nameserver lookups, or registrant referrals. This software was also made publicly available under the terms of an open source license.

The second project was an informal survey of registrants with deployed LDAP servers. This was conducted by using the com, net, org, and edu zone files and testing for the existence of an LDAP server on port 389 using the name of the domain, a host named "ldap" in the domain, and a host named "dir" in the domain (e.g., "foo.com", "ldap.foo.com", and "dir.foo.com"). This survey did not attempt to resolve LDAP services using SRV records in DNS.

The result of this survey found that roughly 0.5% of active domains had an LDAP server. By profiling a server’s root DSA-specific Entry (DSE), the survey found that about 90% of the servers were implementations provided by vendor A, 9% of the servers were implementations provided by vendor B, and 1% of the servers were implementations provided by other vendors. Of the servers queried that were determined to be implementations provided by vendor A, it appeared that about only 10% contained public data (this also led to the assumption that the other 90% were not intended to be publicly queried). Of the servers queried that were determined to be implementations provided by vendor B, it appears that nearly all contained public data.

Appendix B. Acknowledgments

Significant analysis, design, and implementation for this project were conducted by Brad McMillen, David Blacka, Anna Zhang, and Michael Schiraldi. Mark Kosters and Leslie Daigle provided guidance by reviewing this project, the project’s goals, and this document.

Author's Address

Andrew Newton
VeriSign, Inc.
21345 Ridgetop Circle
Sterling, VA 20166
USA

Phone: +1 703 948 3382
EMail: anewton@verisignlabs.com; anewton@ecotroph.net

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.