Network Working Group
INTERNET-DRAFT

S.E. Hardcastle-Kille
University College London
T. Howes
University of Michigan
September 1991

An Access Control approach for Searching and Listing

Status of this Memo

This memo defines an extended ACL (Access Control List) mechanism for the OSI Directory. It is intended to meet strong operational requirements to restrict searching and listing externally, while allowing much more freedom within an organization. In particular, this mechanism makes it possible to restrict searches to certain sets of attributes, and to prevent "trawling": the disclosure of large organizational data or structure information by repeated searches or lists. This capability is necessary for organizations that want to hide their internal structure, or to prevent dumping of their entire database. This memo describes functionality beyond, but compatible with, that expected in the 1992 X.500 standard.

This draft document will be submitted to the RFC editor for information only. Distribution of this memo is unlimited. Please send comments to the author or to the discussion group `<osi-ds@CS.UCL.AC.UK>`.

1 Introduction

This memo defines an extended ACL (Access Control List) mechanism for the OSI Directory. It is intended to meet strong operational requirements to restrict searching and listing externally, while allowing much more freedom within an organization. In particular, this mechanism makes it possible to restrict searches to certain sets of attributes, and to prevent "trawling": the disclosure of large organizational data or structure information by repeated searches or lists. This capability is necessary for organizations that want to hide their internal structure, or to prevent dumping of their entire database. This memo describes functionality beyond, but compatible with, that expected in the 1992 X.500 standard.

The mechanism proposed also allows the Administrative limit to be used for system protection, rather than for access control purposes. This is useful, as the administrative limit is in practice ineffective as a means of access control.

2 Search Control

A search access control is defined, which can be applied to subtree, single level, or base object. This last option is used to protect a specific object from searching, and applies to the object during any type of search.

This ACL controls who may and may not search by certain attributes, and the number of entries that may be returned from a search on the allowed attributes. The normal single entry ACL still controls which information may be returned from each entry.

This ACL approach will work consistently within a single DSA. When a search is traversed over multiple DSAs and a SearchACL is being applied, if a Security Error is returned, it should be assumed that the overall ACL has been violated and a Security Error returned for the whole search.

The ACLs are multiple value, and the most generous applicable should be used.

```
SearchACL ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX SearchACLSyntax
```

```
SearchACLSyntax ::= SEQUENCE {
        who AccessSelector,
         scope BITSTRING {
                subtree(1),
                single-level(2),
                base-object(3) } DEFAULT {subtree, single-level},
        max-results INTEGER,  -- limit for this category
        attribute-selection CHOICE {
           searchable-attributes [1] SET OF AttributeType,
           unsearchable-attributes [2] SET OF AttributeType
           },
        min-key-length-in-substring INTEGER OPTIONAL,
        zero-results-if-limit-exceeded BOOLEAN DEFAULT TRUE
                }
```

The syntax works as follows:

who Controls who the syntax applies to

scope Defines the scope of the access control

max-results Defines the maximum number of results which can be returned.

attribute-selection Searchable attributes can be specified by inclusion or exclusion. The ability to search or not to search the default set of attributes (those not explicitly specified elsewhere) can be specified by use of a null list in unsearchable-attributes and searchable-attributes respectively.

min-key-length-in-substring Gives minimum length to prevent dumping using a* b* in repeated searches. This length should also be applied to filter items which compare string attributes. Where a string range is identified, it must be narrower than min-key-length-in-substring. For example if min-key-length-in-substring is 4, then a range abcde* to abcdf* is acceptable, whereas abcx* to abcy* is not.

zero-results-if-limit-exceeded Only return results if search is precise enough to match less than the limit. This will usually be the case, as it is needed to block intelligent trawling. If zero-results-if-limit-exceeded component is set to TRUE and the max-results limit is exceeded, no results shall be returned and the entire search is abandoned with a security error. If zero-results-if-limit-exceeded component is set to FALSE and the max-results limit is exceeded, partial results are returned just as if a size limit had been reached, and the current portion of the search is abandoned.

Note that there may be a requirement to detect downloading by very large numbers of repeated searches. Mechanisms to prevent this will be investigated in the light of experience.

2.1 Single Level Search

During a single level search, the application of the single level search ACL is straight forward. The ACL of the superior object is used for the duration of the search, even if aliases are dereferenced (i.e., the original ACL is used in this case, not the parent of the object the alias points to if different from the original superior object). SearchACLs with scope base-object in each of the matched entries should be examined.

2.2 Subtree Search

During a subtree search, the application of the subtree search ACL is more complicated, since a subtree search may recurse over multiple sub-subtrees. When a sub-subtree is encountered during the search, its subtree ACL should apply to the elements of the sub-subtree, if it is more restrictive than the currently applicable subtree ACL. Note that this same policy applies when aliases are involved, and the searchaliases option is set. In this case, the currently applicable ACL is taken from the point of the alias, not from the "natural" parent of the aliased subtree as it appears in the DIT. This means that a subtree ACL at the base of a subtree may not be sufficient to protect the component sub-subtrees from unauthorized search.

During a subtree search, if any current search ACL is exceeded, the current portion of the search is abandoned with a security error or a size limit exceeded with partial results, depending on whether the ACL exceeded had the no-results-if-limit-exceeded flag set.

3 List Control

A simplified version of search control is used to control listing. This can be used to control listing at the base object of a list, or to restrict listing of individual objects.

```
ListACL ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX ListACLSyntax

ListACLSyntax := SearchACLSyntax
        (
                WITH COMPONENTS
                        {
                                who,
                                scope (single-level | base-object),
                                max-results
                        }
        )
```

To prevent listing of an individual object, scope should be set to base-object and max-results to zero. This strategy will allow the Admin limit to be used to do system protection, rather than for pseudo ACL purposes.

4 Authentication Policy

In order to make this ACL mechanism useful, we need to address the issue of authentication policy. In validating the who component of the ACLs defined above, we would like to be able to specify different levels of security. For example, a trusting individual might want to believe all names, even those supplied over unauthenticated DSP, while another might want names to be strongly authenticated, and another might view the middle ground of simple authentication as sufficient. The following attribute applies to the single object in which it appears, though it will usually be inherited throughout an entire subtree.

```
AuthenticationPolicy ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX AuthenticationPolicySyntax
        SINGLE VALUE

-- Applies to any ACLs resident in the entry

AuthenticationPolicySyntax ::= SEQUENCE {
        modification AuthenticationPolicy,
        read-and compare AuthenticationPolicy,
        list-and-search AuthenticationPolicy }

AuthenticationPolicy ::= ENUMERATED {
        trust-name (0), -- believe supplied name which must be present
                        -- used if DSP trust chains are OK
        simple (1),
        strong (2) }
```

This attribute is single-valued and applies to all ACLs in the object (including old-stype QUIPU ACLs). To maintain backward compatibility, if this attribute is absent the default shall be to require simple authentication to validate names.

When applied to ordinary QUIPU ACLs, an authentication policy failure causes no access to the entry to be granted. When applied to a search or list ACL, a failure causes the search or list to be abandoned with a security error.

5 Access Selector

The Access Selector is taken from the QUIPU Access Controls. It is repeated here. In a later version of the document, this information will be integrated into an earlier section.

```
AccessSelector ::= CHOICE {
        entry [0] NULL,
                -- DUA identified by the entry
        other [2] NULL,
                -- This indicates 'public' rights
        prefix [3] NameList,
                -- This identifies a prefix name for specified DUAs
                -- e.g. anyone in the UK
        group [4] NameList
                -- For specifying group rights
        }

NameList ::= SET OF DistinguishedName
```

6 Security Considerations

This INTERNET–DRAFT / is concenred with controlling acces to data.

7 Author's Address

Steve Hardcastle-Kille
Department of Computer Science
University College London
Gower Street
WC1E 6BT
England

Phone: +44-71-380-7294

EMail: S.Kille@CS.UCL.AC.UK

Tim Howes
Research Systems
University of Michigan

535 West William St.
Ann Arbor, MI 48103-4943

Phone: +1 313 764 2278

EMail: Tim.Howes@umich.edu