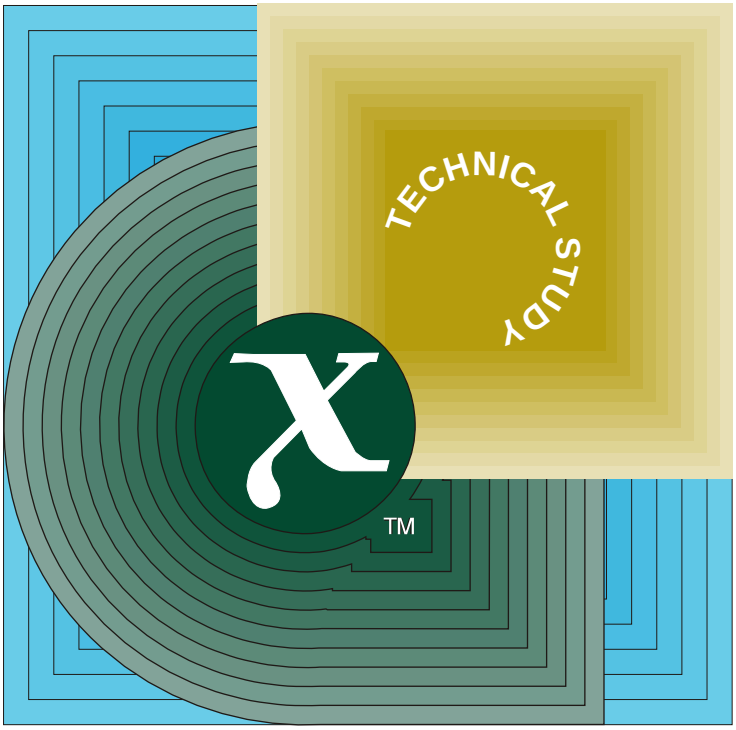
Technical Study

Security in Federated Naming



THE Open GROUP

[This page intentionally left blank]

Technical Study

Security in Federated Naming

The Open Group

Technical Study

Security in Federated Naming

Document Number: E605

Published in the U.K. by The Open Group, February 1997.

Any comments relating to the material contained in this document may be submitted to:

The Open Group
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

or by Electronic Mail to:

OGSpecs@opengroup.org

Contents

List of Figures

Preface

The Open Group

The Open Group is an international open systems organisation that is leading the way in creating the infrastructure needed for the development of network-centric computing and the information superhighway. Formed in 1996 by the merger of the X/Open Company and the Open Software Foundation, The Open Group is supported by most of the world's largest user organisations, information systems vendors and software suppliers. By combining the strengths of open systems specifications and a proven branding scheme with collaborative technology development and advanced research, The Open Group is well positioned to assist user organisations, vendors and suppliers in the development and implementation of products supporting the adoption and proliferation of open systems.

With more than 300 member companies, The Open Group helps the IT industry to advance technologically while managing the change caused by innovation. It does this by:

- consolidating, prioritising and communicating customer requirements to vendors
- conducting research and development with industry, academia and government agencies to deliver innovation and economy through projects associated with its Research Institute
- managing cost-effective development efforts that accelerate consistent multi-vendor deployment of technology in response to customer requirements
- adopting, integrating and publishing industry standard specifications that provide an essential set of blueprints for building open information systems and integrating new technology as it becomes available
- licensing and promoting the X/Open brand that designates vendor products which conform to X/Open Product Standards
- promoting the benefits of open systems to customers, vendors and the public.

The Open Group operates in all phases of the open systems technology lifecycle including innovation, market adoption, product development and proliferation. Presently, it focuses on seven strategic areas: open systems application platform development, architecture, distributed systems management, interoperability, distributed computing environment, security, and the information superhighway. The Open Group is also responsible for the management of the UNIX trade mark on behalf of the industry.

The X/Open Process

This description is used to cover the whole Process developed and evolved by X/Open. It includes the identification of requirements for open systems, development of CAE and Preliminary Specifications through an industry consensus review and adoption procedure (in parallel with formal standards work), and the development of tests and conformance criteria.

This leads to the preparation of a Product Standard which is the name used for the documentation that records the conformance requirements (and other information) to which a vendor may register a product. There are currently two forms of Product Standard, namely the Profile Definition and the Component Definition, although these will eventually be merged into one.

The X/Open brand logo is used by vendors to demonstrate that their products conform to the relevant Product Standard. By use of the X/Open brand they guarantee, through the X/Open Trade Mark Licence Agreement (TMLA), to maintain their products in conformance with the Product Standard so that the product works, will continue to work, and that any problems will be fixed by the vendor.

Open Group Publications

The Open Group publishes a wide range of technical literature, the main part of which is focused on specification development and product documentation, but which also includes Guides, Snapshots, Technical Studies, Branding and Testing documentation, industry surveys and business titles.

There are several types of specification:

- *CAE Specifications*

  CAE (Common Applications Environment) Specifications are the stable specifications that form the basis for our product standards, which are used to develop X/Open branded systems. These specifications are intended to be used widely within the industry for product development and procurement purposes.

  Anyone developing products that implement a CAE Specification can enjoy the benefits of a single, widely supported industry standard. In addition, they can demonstrate product compliance through the X/Open brand. CAE Specifications are published as soon as they are developed, so enabling vendors to proceed with development of conformant products without delay.

- *Preliminary Specifications*

  Preliminary Specifications usually address an emerging area of technology and consequently are not yet supported by multiple sources of stable conformant implementations. They are published for the purpose of validation through implementation of products. A Preliminary Specification is not a draft specification; rather, it is as stable as can be achieved, through applying The Open Group's rigorous development and review procedures.

  Preliminary Specifications are analogous to the *trial-use* standards issued by formal standards organisations, and developers are encouraged to develop products on the basis of them. However, experience through implementation work may result in significant (possibly upwardly incompatible) changes before its progression to becoming a CAE Specification. While the intent is to progress Preliminary Specifications to corresponding CAE Specifications, the ability to do so depends on consensus among Open Group members.

- *Consortium and Technology Specifications*

  The Open Group publishes specifications on behalf of industry consortia. For example, it publishes the NMF SPIRIT procurement specifications on behalf of the Network Management Forum. It also publishes Technology Specifications relating to OSF/1, DCE, OSF/Motif and CDE.

  Technology Specifications (formerly AES Specifications) are often candidates for consensus review, and may be adopted as CAE Specifications, in which case the relevant Technology Specification is superseded by a CAE Specification.

In addition, The Open Group publishes:

- *Product Documentation*

  This includes product documentation — programmer's guides, user manuals, and so on — relating to the Pre-structured Technology Projects (PSTs), such as DCE and CDE. It also includes the Single UNIX Documentation, designed for use as common product documentation for the whole industry.

- *Guides*

  These provide information that is useful in the evaluation, procurement, development or management of open systems, particularly those that relate to the CAE Specifications. The Open Group Guides are advisory, not normative, and should not be referenced for purposes of specifying or claiming conformance to a Product Standard.

- *Technical Studies*

  Technical Studies present results of analyses performed on subjects of interest in areas relevant to The Open Group's Technical Programme. They are intended to communicate the findings to the outside world so as to stimulate discussion and activity in other bodies and the industry in general.

- *Snapshots*

  These provide a mechanism to disseminate information on its current direction and thinking, in advance of possible development of a Specification, Guide or Technical Study. The intention is to stimulate industry debate and prototyping, and solicit feedback. A Snapshot represents the interim results of a technical activity.

Versions and Issues of Specifications

As with all *live* documents, CAE Specifications require revision to align with new developments and associated international standards. To distinguish between revised specifications which are fully backwards compatible and those which are not:

- A new *Version* indicates there is no change to the definitive information contained in the previous publication of that title, but additions/extensions are included. As such, it *replaces* the previous publication.
- A new *Issue* indicates there is substantive change to the definitive information contained in the previous publication of that title, and there may also be additions/extensions. As such, both previous and new documents are maintained as current publications.

Corrigenda

Readers should note that Corrigenda may apply to any publication. Corrigenda information is published on the World-Wide Web at **http://www.opengroup.org/public/pubs**.

Ordering Information

Full catalogue and ordering information on all Open Group publications is available on the World-Wide Web at **http://www.opengroup.org/public/pubs**.

This Document

This document is a Technical Study (see above). It analyses the Federated Naming specification (see referenced document **Federated Naming**) in relation to computer security. In particular, it describes the threats associated with naming services in general and relates these to the referenced Federated Naming Application Programming Interface (API).

Strucure

The document is structured as follows:

- Chapter 1 describes the objective of this paper.
- Chapter 2 provides an overview of the XFN specification presenting the concepts on which that specification is founded as a basis for the subsequent discussion of the security aspects.
- Chapter 3 describes the potential security threats associated with name services.
- Chapter 4 describes the current provisions for security within the XFN API and discusses the potential additional impact handling security may have on the XFN API.
- Chapter 5 proposes some enhancements to the XFN API.

Trade Marks

Motif®, OSF/1® and UNIX® are registered trade marks and the ‘‘X Device’’™ and The Open Group™ are trade marks of The Open Group.

Referenced Documents

The following documents are referenced in this technical study:

Base GSS-API
: CAE Specification, December 1995, Generic Security Service API (GSS-API) Base (ISBN: 1-85912-131-4, C441).

XDSF
: Guide, December 1994, Distributed Security Framework (ISBN: 1-85912-071-7, G410).

Federated Naming
: CAE Specification, July 1995, Federated Naming: The XFN Specification (ISBN: 1-85912-052-0, C403).

Readers may also be interested to refer to:

COAST
: Schuba and Spafford, "Countering Abuse of Name-Based Authentication", COAST Laboratory, Department of Computer Sciences, Purdue University

DNSSEC
: IETF, RFC 2065, Domain Name System Security Extensions.

IDUP
: IETF, Internet-Draft, Independent Data Unit Protection Generic Security Application Program Interface.

IPSEC
: IETF, Internet-Draft, IP Security Protocol.

SSL
: IETF, Internet Draft and Netscape Communications Corporation, Secure Socket Layer (SSL).

Readers may refer to the IETF WWW site (http://www.ietf.org) to access further information on these Internet drafts.

Chapter 1

Introduction

1.1 Scope and Purpose

This document reviews the security aspects of the Federated Naming API specification (XFN) as part of the overall review of the security aspects of all Open Group specifications.

The Security Working Group has produced the Distributed Security Framework (XDSF). This document complements that framework by interpreting the general security considerations addressed by the framework in the specific context of the XFN.

As the XFN is a generic interface to multiple naming systems this document presents an analysis of the security threats and appropriate countermeasures applicable to a general naming service and then considers the impact of these threats and countermeasures in the context of the XFN itself.

The document concludes with specific proposals on how the XFN API may be extended to cater more fully for the security aspects of naming services as a stimulus to promote further discussion.

1.2 Intended Audience

This document is of primary interest to the technical working groups of the Open Group responsible for the Federated Naming Specification and security. It will also be of interest to developers who are implementing or planning to implement a federated naming service based on the XFN.

1.3 Document History

The first draft of this document was reviewed by the OpenGroup Security Program Group at a meeting held in June 1996. This draft includes the comments and observations arising from that review.

The second draft of this document was reviewed by XNET, the Interworking SWG of the Open Group at its meeting in September 1996.

Chapter 2

Federated Naming Specification (XFN)

This chapter provides an overview of the XFN specification for those readers unfamiliar with the specification to place the discussion on security into context. Where possible text and figures have been drawn from the XFN specification itself to avoid errors in interpretation.

2.1 Naming Services

Naming services are a fundamental facility within all IT systems providing the means by which names are associated with objects, and by which objects are found given their names. A naming service provides operations for:

- associating (binding) names to objects
- resolving names to objects, which also identifies how to access them
- assigning information (attributes) to objects and supporting the sharing of that information
- removing bindings, listing names, renaming and so on.

Traditional systems include a multitude of naming services, usually integrated with another service, such as a file system, directory service, database, desktop, mail system, spreadsheet, or calendar.

Each of the naming service interfaces differ widely and the essential naming operations are often obscured.

Furthermore, within a distributed system, the name of an object may be composed of elements from several different naming systems. Such a name is referred to as a *composite name.*

2.2 Federated Naming Specification

A federated naming system is an aggregation of autonomous naming systems that cooperate to offer a standard interface for the resolution of composite names and supports the addition of further types of naming services without requiring changes to applications or to existing member naming systems.

The Federated Naming Specification (XFN) defines an interface comprising a set of common naming operations, that map a composite name to an object reference. The XFN does not specify administrative interfaces as the administrative models of different naming services vary too widely to permit a useful generic treatment.

The XFN is intended to be implemented over a number of existing naming services, using their existing programming interfaces and protocols, as well as with new naming services in the future.

The components of the XFN specification are:

- XFN API
  The XFN API is the means by which clients interface with the XFN services and is described in more detail later in this chapter.

- XFN composite name string representation
  This is not of significance to this report and is not described further. Refer to XFN for further information.
- XFN naming policies
  XFN naming policies are concerned with the structuring of names at differing levels including, global, enterprise and application levels. This is not of significance to this report and the reader is referred to the XFN for further information.
- XFN reference and address
  A reference is the address of an object by which the object may be manipulated. The reference includes an indication of the type of object, and a list of addresses via which it may be accessed. An address comprises an address type identifying the mechanism that should be used to reach the object and an opaque data buffer containing the address information for that mechanism.

There are additionally optional components:

- XFN protocols
  An XFN protocol is a protocol for communication between XFN client and server components and provides for the support of the distribution of XFN services and the interworking between different XFN implementations.
- XFN context implementation
  An XFN context implementation is the implementation of a service interface between an XFN implementation and an underlying name service.
- XFN enterprise policies
  The XFN defines a set of policies for the structuring of common name spaces within an enterprise namespace. These include policies for the structuring of the namespaces for organisational units, hosts, users, filesystems and services. The reader is referred to XFN for further information.

2.3 Implementation Models

The XFN specification does not prescribe any method of implementing the XFN service but does present the following guidelines reproduced from the XFN specification to provide a basis on which to consider the security aspects of XFN.

The three diagrams Figure 2-1, Figure 2-2 and Figure 2-3 serve as examples of the conceptual models of the different possible configurations. The dark shaded boxes shown in these diagrams are building blocks that a naming service integrator needs to provide in order to integrate the naming system with XFN. The modules depicted in the three diagrams are defined as follows:

XFN API
: The *XFN* API is the complete set of interface operations defined in this XFN specification.

XFN Framework
: The *XFN framework* is the implementation of the XFN API, including the client library and the service provider interfaces necessary for integrating native naming systems.

Context Implementation
: The *context implementation* is the naming service-specific module on the XFN client system that is required to integrate legacy naming systems with XFN. The code of the context implementation is a wrapper that maps the XFN API to the API exported by the legacy naming system. The complexity of the context implementation depends on how well the XFN API maps to the native naming service API and which XFN operations are to be supported. At a minimum, the name resolution phase of all operations must be supported.

  The techniques used to access the naming service-specific context implementations from the XFN framework may vary. For systems that support shared libraries and dynamic linking, a common approach might be that the context implementations are dynamically loadable modules.

  This approach of integrating a naming service using a context implementation module does not require any modification to the existing naming service's source code nor does it require access to the naming service's source code. All that is needed is access to the module (library) that exports the naming service-specific API. This approach is by far the easiest and fastest way of adding an existing naming system into the XFN federation.

XFN Client
: The *XFN client* is a module that implements the client protocol machines for the XFN protocols.

  Two XFN protocols are specified, the RPC based protocols for ONC+ systems (specified in RPCL) and for DCE environments (specified in IDL).

  In addition to supporting the protocols, the XFN client might provide services typically offered by naming clients, such as caching. The extent of this support is implementation-specific.

XFN Protocol Exporter
: The *XFN protocol exporter* is the module required on systems that export one of the XFN protocols. This could be a new naming system, an existing naming system that was modified to also support XFN protocols, or a system that supports the XFN client library and also exports XFN protocols (capable of acting as surrogate client).

  The advantage for naming systems that support one of the specified protocols is that any existing XFN client that imports the protocols can be used to communicate with it. This is particularly useful for applications that need to export naming interfaces. Application programmers do not have to duplicate the client-side implementation and they do not have

to invent new naming interfaces. This provides additional benefits such as the ability to utilise caching and other mechanisms provided by the XFN client implementations, and a direct (and possibly more efficient) mapping of XFN operations to the application's naming operations.

Figure 2-1 shows the layering of the XFN client library on top of existing naming system clients on the same system. None of the legacy naming systems needs to be modified.

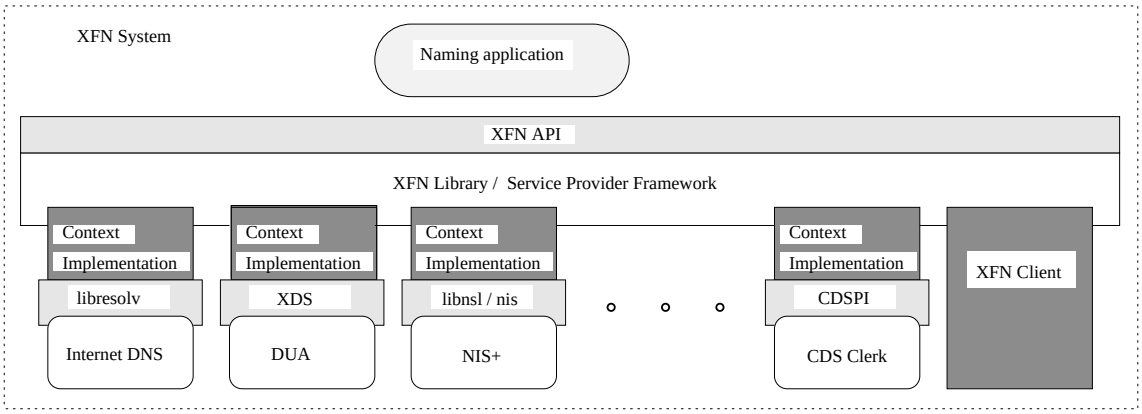


Figure 2-1 XFN Configuration using Client Context Implementations

Figure 2-2 shows multiple XFN systems that are connected via one of the specified XFN protocols. The client in this picture is a lightweight XFN client. The servers shown are name servers that directly export one of the specified XFN protocols.

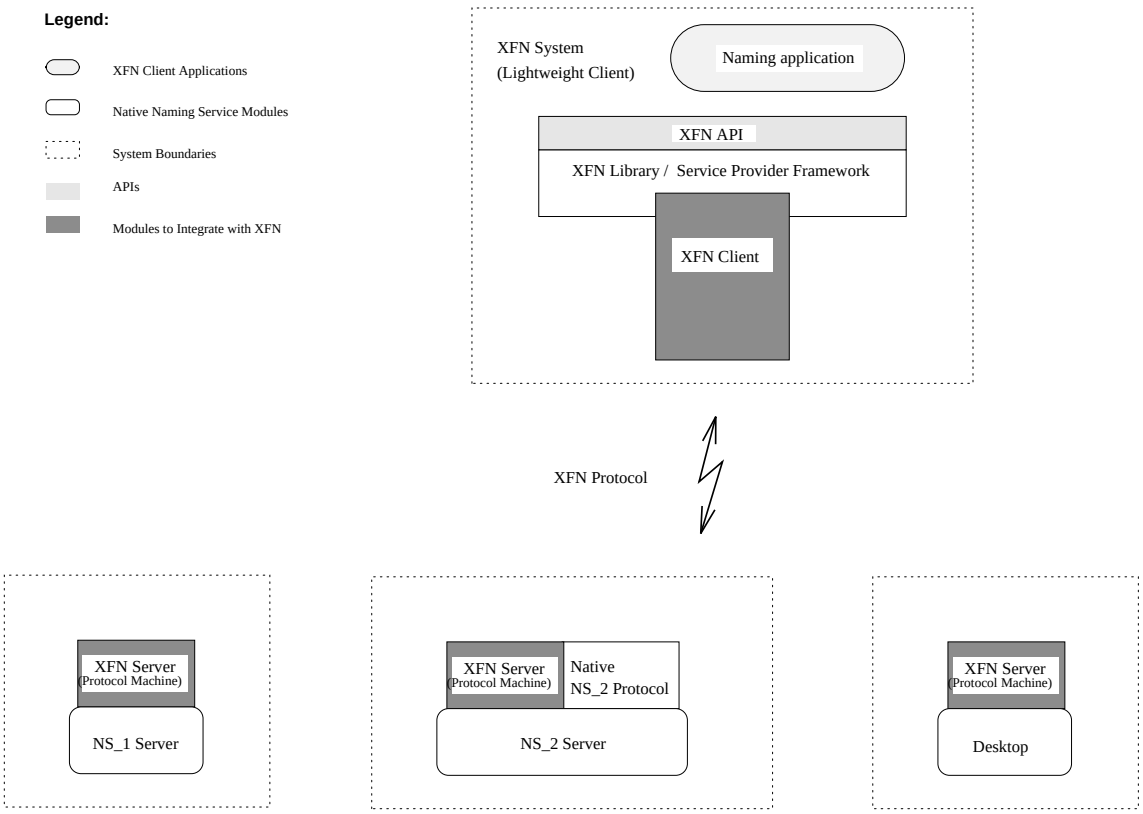


Figure 2-2 Lightweight XFN Client Configuration

The two modules shown in Figure 2-3 are a lightweight XFN client and a server that acts as an intermediary. Similar to the client in Figure 2-2, the client in Figure 2-3 is a truly lightweight XFN client. None of the legacy naming system clients needs to be installed at that system. Depending on the client system's requirements, the XFN client can be implemented and configured to consume more or less resources, determined based on needs and availability. The XFN client might simply defer to mechanisms (such as for caching and replication) provided by the native naming system clients.

The legacy naming system clients in Figure 2-3 reside on a remote system (similar to Figure 2-1) that also exports at least one of the XFN protocols. This remote client can be viewed as a surrogate or proxy client that acts on behalf of the initial requestor and performs the native naming system functions.

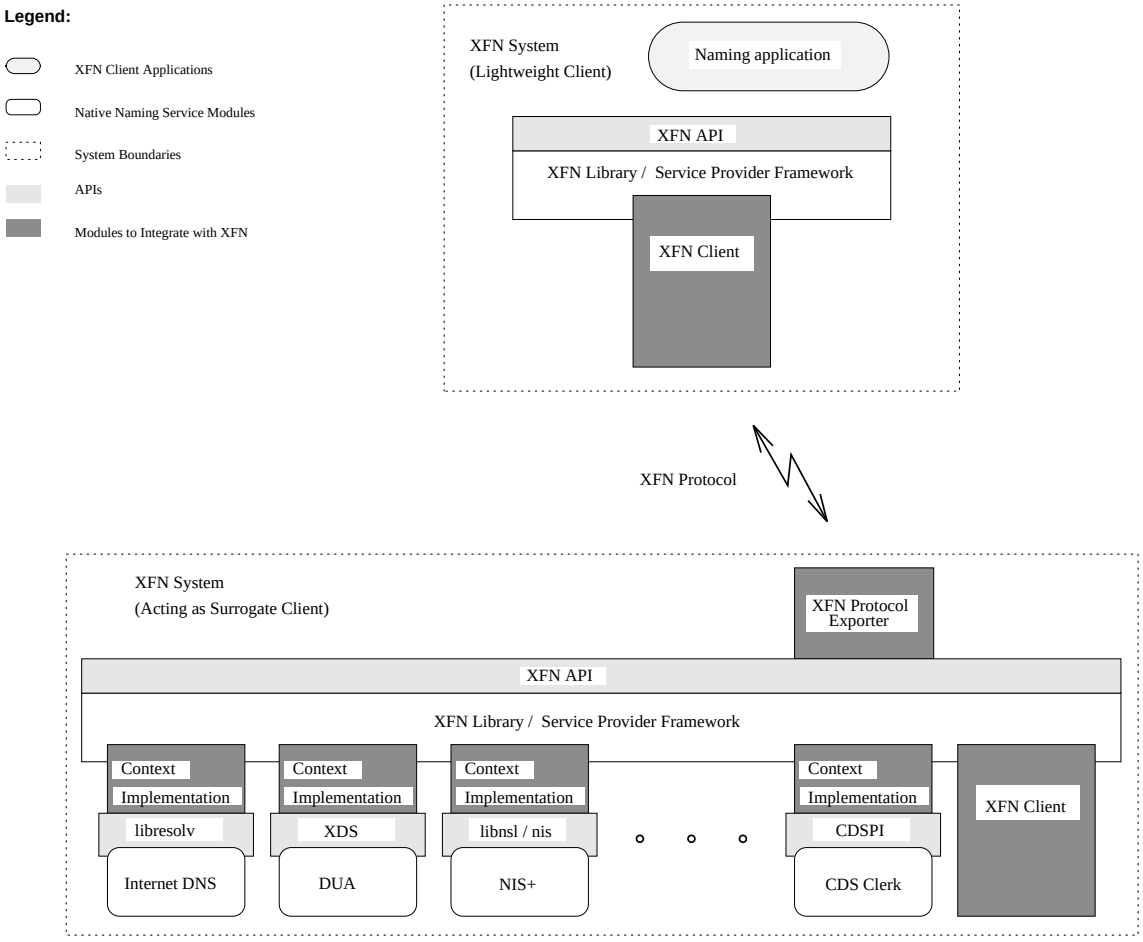


Figure 2-3 XFN Configuration with Surrogate Client

Another aspect shown in Figure 2-1 and Figure 2-3 is the capability of the surrogate client to also import the XFN protocol (*XFN client* module). Such a configuration could serve emerging XFN servers or existing name servers that export one of the specified XFN protocols in addition to, or in replacement of, the native protocol.

Note that a *context implementation* precisely defines the set of modules that are co-located with the XFN framework to map the XFN API to the native naming service API. However, in the context of this description, we also use the term *context implementation* to mean the XFN

mapping code that is necessary at the server of a naming system that directly exports one of the XFN protocols (*XFN Protocol Exporter* and *XFN Server* in the diagrams).

2.4 XFN API

The XFN client interface comprises the following categories of interfaces:

- the basic context interface
  The basic context interface provides the operations for naming, such as binding a name to a reference, looking up the reference bound to a name, unbinding a name.
- the basic attribute interface
  The basic attribute interface provides operations to examine and modify attributes associated with named objects.
- the extended attribute interface.
  The extended attribute interface provides operations to do searching and creation of objects in the namespace with attributes.

A summary of the XFN functions is presented below with the functions grouped in a categorisation with a security perspective in mind. That is functions are grouped with the definition of security policy in mind. The names of the functions listed provide a good indication of their purpose and the style of the interface. The reader should refer to XFN itself for more detail.

All name resolution operations occur by reference to a *context*. A handle to an XFN context is an input parameter to almost every XFN function. Before the XFN functions may be used an initial XFN context handle must be obtained. Handles to additional contexts represented by an XFN reference are obtained from a name lookup operation. The functions that manipulate XFN contexts are:

- *fn_ctx_handle_from_initial*( )
- *fn_ctx_handle_from_ref*( )
- *fn_ctx_get_ref*( )
- *fn_ctx_get_syntax_attrs*( )
- *fn_ctx_handle_destroy*( )

The principal purpose of a name service is to resolve names to object addresses, or XFN reference. The functions that support this purpose are:

- *fn_ctx_lookup*( )
- *fn_ctx_list_names*( )
- *fn_namelist_next*( )
- *fn_namelist_destroy*( )
- *fn_ctx_list_bindings*( )
- *fn_bindinglist_next*( )
- *fn_bindinglist_destroy*( )
- *fn_ctx_lookup_link*( )

To support the name resolution process name to object bindings must be created and managed. the set of functions that support this functionality are:

- *fn_ctx_bind*( )
- *fn_attr_bind*( )
- *fn_ctx_unbind*( )
- *fn_ctx_rename*( )
- *fn_ctx_create_subcontext*( )
- *fn_attr_create_subcontext*( )
- *fn_ctx_destroy_subcontext*( )

The creation of a subcontext creates a new branch in a context namespace. (It is like a *mkdir* operation in a filesystem.)

The attribute related functions included under bind management are part of the extended attribute interface and provide for the creation of objects with attributes assigned as part of the creation operation.

A secondary purpose of a name service is to support the assignment of information attributes to objects and the sharing of that information between users of the name service. The XFN includes functions that operate with single attribute values, multiple attribute values, and multiple attributes:

- *fn_attr_get*( )
- *fn_attr_modify*( )
- *fn_attr_get_values*( )
- *fn_valuelist_next*( )
- *fn_valuelist_destroy*( )
- *fn_attr_get_ids*( )
- *fn_attr_multi_get*( )
- *fn_multigetlist_next*( )
- *fn_multigetlist_destroy*( )
- *fn_attr_multi_modify*( )

The extended attribute interface provides for search capability based on attributes. The functions are:

- *fn_attr_search*( )
- *fn_searchlist_next*( )
- *fn_searchlist_destroy*( )
- *fn_attr_ext_search*( )
- *fn_ext_searchlist_next*( )
- *fn_ext_searchlist_destroy*( )

Chapter 3

Name Service Threats and Countermeasures

This chapter outlines the generic threats to IT systems and describes the nature of a name service architecture to set the context for the subsequent discussion of name service specific security concerns, specific threats and appropriate countermeasures.

3.1 Generic Threats

The Distributed Security Framework (XDSF) categorises the generic threats to IT systems as follows:

- unauthorised modification
- unauthorised disclosure
- unauthorised use of resources
- unauthorised denial of service
- false repudiation.

These generic threats are realised by specific actions or sets of actions. For example, eavesdropping on a communications channel may reveal user passwords. Those passwords may then be used to enable the user who eavesdropped to masquerade as other users. The XDSF describes the many ways in which the generic threats above may be realised and also describes generic measures that may be deployed to counter these threats. This chapter identifies the threats that are specific to a name service.

As well as threats arising from unauthorised operations threats also arise from the irresponsible or malicious exercise of authorised operations. It may not be possible to prevent threats arising from authorised operations. however, by recording and analysing system operations it may be possible to deter such behaviour, or react and recover promptly to its occurrence.

3.2 Name Service Architecture

A name service is typically implemented as a client-server architecture, as described in Chapter 2, and may be considered to have the following aspects from a security viewpoint:

- **Data Storage**
  The bindings between names and addresses and the attributes of objects managed via the name service have to be stored. These are generally stored in databases controlled by services that may be authorative or non-authorative for a particular binding. Object attributes may be stored by the name service or by the object itself.

  An authorative service for a binding is a service that has management responsibility for the creation, modification, and deletion of the binding. An authorative service generally stores binding information on a permanent basis. A non-authorative service does not have management responsibility for a binding and is generally caching the information as a result of a previous enquiry for performance purposes. That is, a non-authorative service generally stores binding information on a transient basis.

  A name service client may request that the binding information supplied is authorative.

- **Communications**
  A name service uses communication services between its components and possibly between client applications and its components. In the case of communications between XFN clients and an XFN protocol exporter then an RPC based protocol is specified.
- **Client and Administrative APIs**
  Access to name services are provided via interfaces that may be invoked by client programs. Additionally, interfaces may be provided for the administration of the name service.

3.3 Name Service Security Policy Concerns

3.3.1 Availability

Name services are a critical component of distributed systems providing a service via which users and applications may locate objects with which they require to interact. The availability and responsiveness of name services therefore have a direct impact upon the overall usability and responsiveness of a whole system. An attack on the availability of a name service may be used as the basis of a denial of service attack on a system or other individual services.

3.3.2 Integrity

The maintenance of the integrity of the binding of a name to an object is a fundamental requirement of a naming service. The modification or destruction of a binding between a name and an object may lead to attacks on other system services through denial of service or through masquerade. Denial of service can arise from an inability to locate a service, masquerade attempts can occur through deliberate modification of a binding or interference with the name resolution process resulting in a caller interacting with a subverted object.

Associated with the integrity of the binding of a name to an object, a security policy may require uniqueness of the binding of a name to an object. A name should only refer to a unique object. If a name does not refer to a unique object then any reliance upon that binding, for example in support of authentication, is compromised. The existence of multiple bindings to a single object may or may not be a security concern.

This requirement for uniqueness may be extended in time to also cover the reuse or re-assignment of names. This is of particular security significance in the context of the historic analysis of security audit trails for which the binding information needs to be preserved. Note that a binding may actually reference multiple instances of a single object, or a distributed object, for the purposes of resilience.

Also note that in some namespaces the re-assignment of names is a fundamental part of the operation of applications. For example, in the filesystem namespace a wordprocessing application frequently updates a document by creating a new object, a file, renaming the original file as a backup and re-assigning the document name to the new object. In this example the uniqueness in the binding is between a name and a user concept of a document. The mapping between the document object and a filesystem object varies according to a policy enforced by the appropriate application.

If the name server also supports the assignment of attributes to objects then the integrity of that assignment may also be significant if the attributes are used for security related purposes.

3.3.3 Confidentiality

In the context of a naming service, confidentiality is generally of lesser importance than integrity and availability. It is of significance if the revealing of the existence or location of objects or of attributes assigned to objects is subject to control under the applicable security policy.

An example may be an enterprise policy prohibiting disclosure of information about the enterprise's user and service namespaces to principals that are not part of the enterprise.

3.3.4 Accountability

A security policy is likely to require principals that modify binding and attribute information to be held accountable for their actions. This is often likely to particularly apply to the principals with administrative authority over a name service. Accountability is generally enforced through the use of a security audit service to record security significant events and their subsequent analysis.

3.4 Data Store Threats

Destruction or corruption of a name server's database may be used to attack the availability of a name service. That is, unauthorised modification can lead to denial of service.

Unauthorised modification of a name server's database may be used to attack the integrity of the name service.

Read access to a name server database may be used to attack the confidentiality of name service information, that is unauthorised disclosure.

Threats and appropriate countermeasures include:

- **Back Door Interfaces**
  Operating system file system services, backup/restore facilities, and low-level disc access facilities may be used to read or modify data held on disc, potentially bypassing any application level controls on such access.

  Appropriate countermeasures depend upon the existence of suitable operating system access controls, physical access controls and administrative procedures in the processing environment supporting the name service components.

  Additionally, cryptographic techniques may be used by the application to protect its data by sealing or signing or both the data whilst stored within operating system resources. The security of such techniques then depends upon the protection provided to the cryptographic keys used to provide those services.

- **Reuse of Resources**
  Information can be disclosed or corrupted if the resource in which it is stored is reused for other purposes without the previous contents being purged.

  Reliance is generally placed upon an operating system to purge memory before allocation to a process. Responsibility rests with the application developer to purge any memory under the direct control of the application that is reused.

- **Physical Damage**
  A name service database may be destroyed or corrupted when the physical media on which it is stored is damaged. This can arise from physical damage or by electromagnetic means.

  Appropriate programmatic countermeasures include the provision of replicated name servers. Administrative countermeasures include normal business continuity measures such

as backup procedures and service recovery plans. High availability servers could also be used.

- **Resource Overload**
  If a disc or other resource such as memory is exhausted then a name server may not be able to store new bindings and object attributes and may not be able to process requests for name resolution. This may arise from excessive requests upon the name server itself or by the overloading of another service that is competing for resources with the name service.

  Appropriate countermeasures include resource level monitoring and administrative alarms.

3.5 Communication Service Threats

Interference with communication services offer the opportunity to affect the availability, integrity and confidentiality of name services.

Threats and appropriate countermeasures include:

- **Masquerade of Communications Partner**
  Masquerading as a name server enables an attacker to supply modified name and object bindings. Such an attack may subvert the name service for a considerable period if such false information is cached by a name service client.

  Masquerading as an authorised client may enable an attacker to create new, or modify or delete existing bindings of names and attributes to objects.

  These threats are countered by the use of appropriate authentication exchanges between client and server components. These services may be invoked via the GSS-API or other security protocol, for example SSL or IPSEC.

- **Message Insertion/Modification**
  An attacker may insert or modify messages into the communication stream so that it appears to a name service client that the name server has returned more information than has actually been requested. Such additional information may be cached by the name server client and subsequently used in name resolution or attribute retrieval operations.

  This may be countered by a client verifying that the information returned is only that requested and by the use of cryptographic techniques such as digital signatures for verifying the integrity and origin of messages received.

- **Tampering with Communications Service Configuration**
  Access to name system services may be disrupted by modification of the configuration of the communications services, at either the client or server hosts. This may also be used to masquerade as a communications partner by another host.

  This may be countered by regularly monitoring the communications service configuration on both server and client hosts or by signing the configuration files themselves and verifying the signature whenever accessed by the name service.

- **Eavesdropping on Communications Traffic**
  Eavesdropping on communications traffic may be used to obtain unauthorised disclosure of name service information, such as object attributes.

  This may be countered by the enciphering of messages exchanged between clients and servers or between servers themselves. These services may be invoked via the GSS-API or other security protocols, for example SSL or IPSEC.

- **Network Damage or Congestion**
  The availability of a name service may be impaired by overloading the network with spurious traffic or physically disrupting the network. Also an individual name server may be saturated with spurious client requests so that it is unable to respond to valid client requests.

  The saturation of a name server may be countered in some part by applying quotas to the number of requests of specific types (some of which take longer than others) a server will concurrently service from a single client. However, it is not possible to counter an overloading with spurious connection requests.

- **Repudiation**
  The originator of a message that modifies a name to object binding, or that modifies object attributes, may deny having sent it.

  This may be countered by the use of message origin authentication based on digital signatures and by security auditing of the use of binding and attribute management functions.

3.6 Analysis of Name Service Requests

An analysis of name service requests may reveal information about a client's activities, for example which trading partners it is currently dealing with. This may be countered to some extent by ''traffic padding'', that is, generating false requests to mask the real reqests.

3.7 Programming Interface Threats

Threats and appropriate countermeasures include:

- **Masquerade**
  A name service client can masquerade at a programmatic interface by presenting a user identity that does not belong to its principal (normally the user that invoked it).

  This may be countered by requiring a client to authenticate itself, or its ultimate principal, to the name service.

- **Exercise of Unauthorised Authority**
  A program may exercise authority for which it is not authorised by attempting, under the identity of its own principal, to use services that it is not authorised to use, or attempting to access information that it is not authorised to access, perhaps claiming privileges to which the client is not entitled.

  This may be countered by the enforcement of an access control policy based upon authentication and authorisation services.

- **Abuse of authorised access**
  Authorised access either via the normal name service client interfaces or via name service administrative interfaces may be abused, for example by performing actions irresponsibly.

  Appropriate countermeasures rely upon the definition of operating procedures for the use of the service and the auditing of activity in support of making the initiators of actions accountable for those actions.

- **Substitution or Modification of Name Service Programs**
  Name service programs may be substituted or modified to include trojan horses. By this means Name service requests and responses may then be modified.

  This may be countered by the control and monitoring of the software configuration of hosts.

- **Substitution or Modification of Dynamic Linked libraries**
  The substitution of dynamic linked libraries used to implement name server specific context implementations may result in the invocation of subverted name service clients or the interception and modification of name service requests and responses.

  This may be countered by the use of static linking or by regular monitoring of host system configuration.

- **Incomplete Operations**
  The interruption of a binding or attribute management operation may result in a partially completed update compromising the integrity of that aspect of the name service. This threat may be particularly significant when object attributes manipulated via a name service interface are actually located with the object itself and managed by the object management system.

  An appropriate countermeasure is for an implementation to use transaction processing techniques to ensure the atomicity of operations.

3.8 Generic Countermeasures

The following countermeasures do not relate to individual threats but form part of an overall protective environment:

- **Distributed Security Services**
  Distributed security services are security services deployed within a distributed environment which are trusted by other system components. Examples include an authentication service, security attribute service, key distribution service, and a certification authority service. The use of a combination of these services can be used to support communication security services and access control enforcement in the distributed environment across and between the other system components.

- **Event Detection**
  This is the detection of events relevant to security including actual and attempted policy violations and service availability.

- **Security Audit**
  The recording of security relevant events does not in itself prevent breaches of security but it does provide a means of holding individuals and organisations accountable for their actions and therefore can act as a deterrent.

  Analysis of a log of information requests upon a name server may reveal information about the source of potential attacks on the enterprise's systems.

- **Security Recovery**
  This is the taking of appropriate action to recover from actual or attempted security violations. An example is the refreshing of a cache of name service information if a component suspects a security compromise. Security recovery actions may be automatically initiated by an event management service in response to detected events.

3.9 Access Control within Name Services

Name services are frequently integrated with other services, such as a filesystem, directory service, etc. As such there is no single generic access control policy applicable to all instances of a name service. However, it is possible to discuss some commonly applicable policy types, and these are addressed in this section.

3.9.1 Context Functions

These may be subject to an access control policy if they require name resolution to be performed. See Name Resolution functions below.

3.9.2 Name Resolution Functions

Name resolution functions are often offered to anonymous principals. For a significant set of name services, such as DNS, the identity of an individual user is irrelevant for access control policy enforcement and accountability is not a concern. What is more likely to be relevant for access control purposes is the enterprise or organisational unit affiliation of a user. In this case the user principal does not require to be authenticated to the name service but a client may have to present a security attribute representing the enterprise or organisational unit. The name server needs to be able to verify the authenticity of that security attribute.

In some services, for example a file system, name resolution functions such as listing the names within a context are subject to a similar access control policy to the access control policy that applies to access to the objects themselves.

3.9.3 Binding and Attribute Management Functions

In general a name service security policy is concerned with the accountability of individual user principals for the use of functions that create, modify, an destroy bindings and object attributes. The support of such accountability requires the authentication of individual user principals and the propagation of that information between the name service components participating in the operation.

The access control policy enforced on binding and attribute management may be based on a concept of the possession of authorities for the execution of a subset of a name services functions or on the basis of a relationship to the object, for example ownership or specifically assigned access rights.

Possession of authority is frequently linked with administrative responsibility for aspects of the name service and may be represented by a capability or privilege mechanism.

Chapter 4

Implications of Security on XFN

4.1 Mapping of XFN to XDSF Concepts

A basic concept of the Distributed Security Framework (XDSF) is to view an IT system as comprised of a set of interacting and in some cases interdependent security domains. A security domain is defined as " A set of elements, a security policy, a security authority and a set of security-relevant operations in which the set of elements are subject to the security policy, administered by the security authority, for the specified activities". At this level each service (or application) can be viewed as an individual security domain enforcing its own security policy over the application entities within the domain. Each service or application services principals, that is users or other services, that are external to the service or application.

Services are provided by the exchange of information across the domain boundary. The service can only identify the principal via information exchanged between the principal and the service. In order to prevent the masquerade of one principal by another the identification information supplied requires authenticating.

The service may exercise access control on the basis of the authenticated identity, or it may associate other security attributes with the identity for the purposes of mediating authorisation to access entities or exercise activities of the service. For example security attributes may include group affiliation, authorities or privileges, role, etc.

Figure 4-1 illustrates the security domains involved in the implementation of XFN. Each name service that is integrated as part of XFN comprises a security domain in its own right. The entities within the domain are the bindings and object attributes maintained by the name service for the namspace that it services. Each name service enforces a security policy over the operations it provides to access and maintain those bindings and attributes in that namespace.

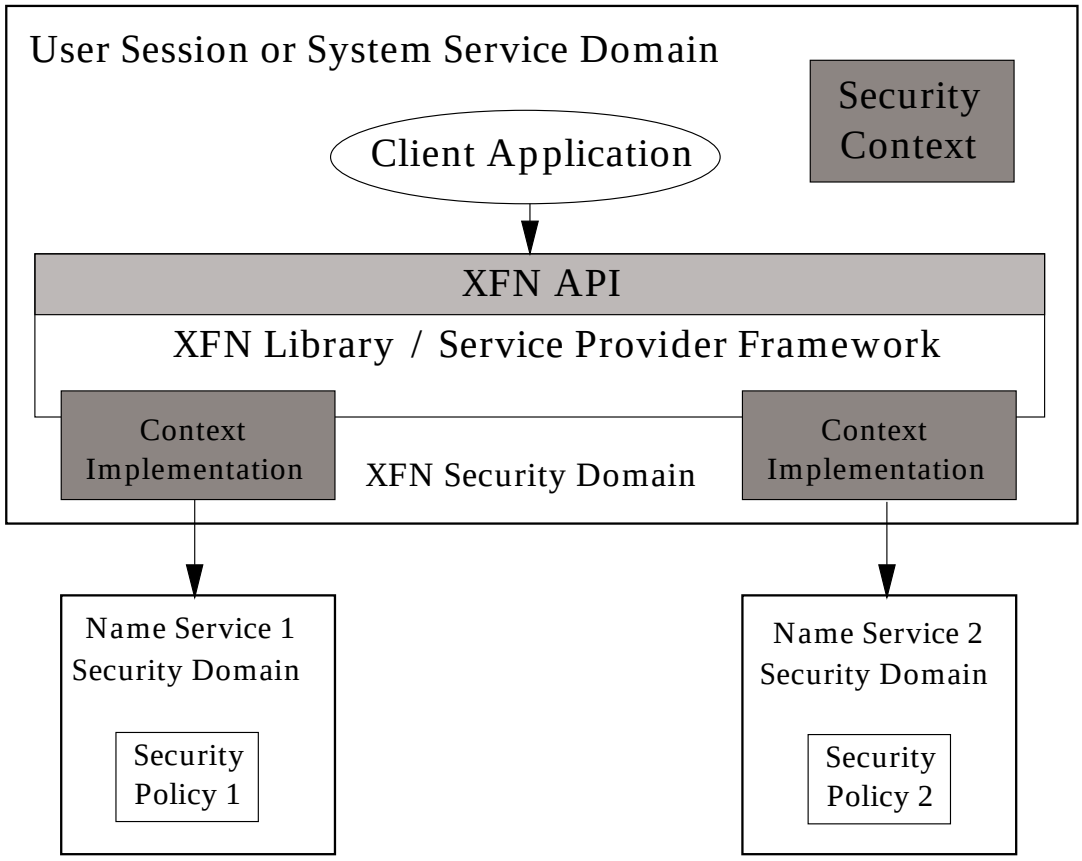


Figure 4-1 Security Domains in XFN Operation

The application client invoking the XFN API executes within a security domain representing an end user principal or a system service principal. Associated with the application client is a security context that includes the authenticated identity of the principal and any security attributes associated with the principal within that domain.

In invoking the services of a name server the XFN implementation must propagate the appropriate parts of the security context, normally at least the principal identity, to the underlying name service. The name service will require to verify the authenticity of the information it is passed.

In the context of the XDSF, the distributed security services provide an authentication service and security attribute service that are trusted by both the user sign-on service and the name service. The name service will verify the authenticity of the security context information it is passed by verifying that it is originated by or certified by the security services it trusts.

4.2 Relevant Threats

Of the threats identified in the previous chapter the following are not directly relevant to, or cannot be addressed at the level of the XFN API Specification:

- data store threats arising from security breaches on a name server host
- denial of service through physical damage to a name service data store
- disclosure or modification through reuse of resources
- resource overload other than that arising through the XFN API
- denial of service arising from network damage or congestion
- disclosure by deduction, for example traffic analysis
- tampering with Communications Service Configuration
- eavesdropping on communications traffic
- substitution of name service programs or dynamic linked libraries
- incomplete binding and attribute management operations.

Even though these threats are not directly relevant to the XFN specification itself they are of relevance to name service implementations and could be included in any guidance to implementors.

The threats that are directly relevant in the context of the XFN API itself are:

- masquerade as a communications partner
- message insertion or modification
- repudiation
- masquerade of a user principal
- exercise of unauthorised authority
- abuse of authorised access
- resource overload via the XFN API
- non-availability of name server.

4.3 Current Security Provisions within XFN API

The XFN API includes the following statements regarding security:

> XFN does not define a security model nor a common security interface for contexts. Security-related operations, such as those required for authentication or access control, fall into the category of administrative interfaces which are expected to differ widely amongst federation members. Any single choice of security model at the XFN layer is prone to be fundamentally incompatible with the security provided by other direct interfaces, thereby giving rise to inconsistent protection that is susceptible to compromise. Consequently, the security administration interface is kept entirely separate from the federated naming service interface, outside the scope of XFN.
>
> The XFN does, however, provide means by which security can be integrated with specific XFN implementations. Operations that fail due to security-related problems can indicate in the status code the nature of the failure.

Authentication

> Authentication is handled in the modules that implement the service interfaces for each particular member naming system. It occurs as part of the communication that occurs between the client and the naming service. Significant engineering issues remain when multiple security mechanisms and administrative domains are involved. These problems can be addressed on a one-to-one basis, or via a general federated security model, outside the scope of XFN. XFN provides the status code [FN_E_AUTHENTICATION_FAILURE] to indicate that an operation failed due to authentication errors.

Access Control

> Given the ability to authenticate the principal making a service request, a context service provider must then decide whether this principal should be granted or denied the request. Access control is to be handled through additional interfaces in the specific contexts. This can be done by having operations that control default authorisation at context creation and binding creation times, and having interfaces that modify the current authorisation settings. This is analogous to the *umask* and *chmod* scheme for POSIX.1 files.
>
> The access control model is outside the scope of XFN.
>
> XFN provides the status codes [FN_E_CTX_NO_PERMISSION] and [FN_E_ATTR_NO_PERMISSION] to indicate that an operation failed due to access control errors. In the case that the principal is not authorised to know that access has been denied due to permission problems, the status code [FN_E_NAME_NOT_FOUND] or [FN_E_NO_SUCH_ATTRIBUTE] is returned.

As indicated in the previous chapter, this Study generally concurs with the above approach. A name service may support anonymous client principals, or exercise access control at the levels of an enterprise, organisational unit, or individual. It is necessary for an implementation of the XFN or underlying name services to be able to retrieve and propagate any necessary authentication and security attribute information to support the particular access control policies enforced by a name service but it is not necessary to supply or manipulate such information via the XFN API. See Section 4.4 for an expansion of the potential impact of security on the XFN API.

It should be noted in the guidelines on implementation that there is an implicit assumption that the *context implementations* are able to retrieve appropriate authentication credentials from the processing environment of the client application. This may be via operating system process attributes or retrieval from a suitably protected credential cache perhaps combined with the invocation of appropriate distributed security services.

Any further discussion on this point would result in a general dissertion on distributed system security services for which the reader is referred to the XDSF.

The XFN specification includes the following measures to address the non-availability of services:

- a status return **FN_E_INSUFFICIENT_RESOURCES** may be returned by XFN functions. This can be used both for actual resource exhaustion and also if an implementation enforces a request quota system
- an XFN reference includes the provision for multiple addresses thus supporting the use of replicated services.

4.4 Potential Impact on XFN API

The services offered by XFN API may be categorised as:

- **Information Retrieval**

  Information retrieval services include name resolution and attribute enquiry.

  The principal concerns with these services are availability and the integrity of the returned information. Availability is an issue for the implementation of an XFN service. The integrity of the information returned requires the authentication of the name server providing the information to the client.

  The XFN API already includes accommodation for issues of service availability through the capability to support multiple addresses in a XFN object reference thereby supporting replication of services.

  The XFN API supports the concept of *authoritive* contexts but this is with emphasis on the time currency of cached name resolution information rather than its authenticity in terms of origin and integrity. Proposals are detailed in the next chapter to also accommodate authenticity of name resolution information in the XFN API.

- **Name and Object Binding Management**

  These include the operations of creating, deleting, and modifying a binding (renaming).

  The principle concerns with these services is the maintenance of the integrity of the binding information.

  Protection of the integrity of the binding information requires the enforcement of an access control service based upon authentication of the client and verification of appropriate authorities (or privileges) in the context of the specific name service. The authorities required will vary according to the name system being manipulated.

  A further concern is the atomicity of the binding operation. The atomicity of the XFN operations is dependent upon the atomicity of the underlying name service operations. An appropriate conformance requirement is to require an implementator of a context implementation to state which XFN operations are atomic or not when invoked over the context implementation.

  Binding operations are potentially security significant events and as such should result in the generation of an audit event. This is discussed further in the next chapter.

  An additional security policy concern may be constraints on the re-use or re-assignment of names to different objects. This may have particular impact on the historical analysis of security audit trails for which the binding information has to be preserved as well operational impacts. Proposals are included in the next chapter to accommodate the impact of name re-use policies on the XFN API.

- **Attribute Management**

  These include adding, modifying, and deleting attributes. In the case where the name service maintains the attributes in association with the name, the name service enforces access control policy.

  In the case when the attributes are maintained with the object named by the naming service, the object management service enforces the access control policy and the name service may have to operate with delegated authority from the original requesting principal.

  Attribute management operations are potentially security significant events and as such should result in the generation of an audit event. This is discussed further in the next

chapter. No other changes to the XFN API are thought necessary to accommodate attribute management access control policy enforcement.

- **XFN Policies - Access Control Extension**

  XFN identifies some particular common namespaces, namely enterprise, organisational units, hosts, users, file system and services and presents some naming conventions for these.

  As a discussion point it may be appropriate to identify common access policies for these and define specific XFN Authorities to exercise sets of XFN functions for each of the XFN policy namespaces. As an example, the current work on account management within the Security Program Group proposes the following example set of authorities for the user name space:

  — **XSSO_USER_ADMIN**
  Required to create and delete user accounts.

  — **XSSO_USER_AUDITOR**
  Required to modify user account audit service attributes.

  — **XSSO_USER_SECURITY**
  Required to modify user account credentials and to enable and disable user accounts.

  — **XSSO_USER_ATTRIBUTES**
  Required to modify user account attributes that are not controlled by XSSO_USER_AUDITOR or XSS_USER_SECURITY authorities.

  — **XSSO_ATTRIBUTE_ADMIN**
  Required to manage the attribute set applicable to the user name space. That is, create or delete new attributes, assign the requisite authorisation for their assignment to users.

4.5 Potential Impact on XFN Protocol

XFN defines two XFN export protocols, one based on DCE and the other based on ONC. DCE RPC by definition operates over communication services that incorporate distributed security services that provide support for authentication and message integrity and confidentiality services. ONC RPC can be used with a number of authentication flavours. An ONC RPC authentication flavour based on the GSS-API over a Kerberos mechanism is available and is being discussed within the IETF currently for standardisation. This ONC RPC authentication flavour provides equivalent support for authentication and message integrity and confidentiality services.

The use of such services is implemented below the RPC interface seen by applications and no change is required to the XFN RPC protocol to specifically address the use of these security services. This is illustrated in Figure 4-2.

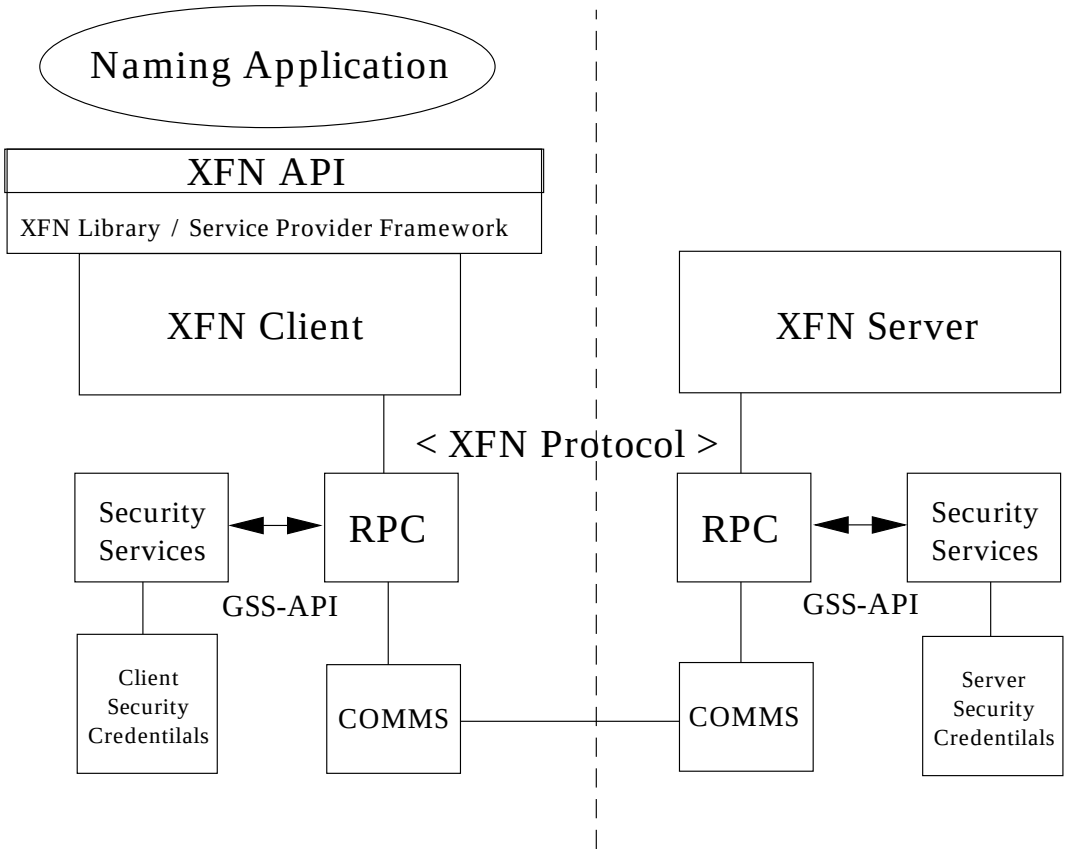


Figure 4-2 Security and XFN Protocol

The specification should include reference to these security services within the guidelines to implementors.

Chapter 5

Proposed Enhancements to XFN

5.1 Name Resolution Integrity

A principal security concern of a name resolution service is the integrity of the name to object binding information returned from a name server. This requirement can be addressed by the incorporation of message origin authentication and message integrity services by a name server implementation. An example of both services is the inclusion of digital signatures with all name resolution responses by a name server. A name service client can verify both the origin and the integrity of the response by verifying the digital signature.

The availability of message origin authentication and message integrity services to a client depends upon their implementation by a name server, and not all name servers will offer this service. Whether message origin authentication and message integrity services are actually used to verify the integrity of name resolution responses is at the option of the name service client. A requirement to use these services could be configured by an installation as enterprise policy, or it could be provided as a caller option.

Integrity is associated to the concept of *authoritive* information which is already incorporated into the *fn_ctx_handle_from_initial*( ) and *fn_ctx_handle_from_ref*( ) functions as a caller controlled option. However, the concept of *authoritive* information as currently expressed in XFN is more concerned with time currency than authenticity. Currency of information and authenticity are different aspects of integrity. A client may be concerned with the authenticity of the information it uses independently of its currency. Techniques such as digital signatures mean that the authenticity of cached information may be verified, however old it is. If verification of currency is required then a time stamp could be incorporated in the reference returned and included in the data under the digital signature.

XFN works above other name services. XFN cannot therefore enforce a requirement for message origin authentication and message integrity on name services that do not support it. However, it should support the use of such services when they are available and ensure that its own XFN protocols include such services.

The following suggestions are made to promote discussion:

- A status code **FN_E_INTEGRITY_FAILURE** could be returned if a name resolution integrity failure is detected when such a check is required by policy for all, or some, of the name services invoked to resolve a composite name.
- A status code **FN_E_NO_INTEGRITY_CHECK** could be returned if client policy requires name resolution integrity checks but one or more of the name services involved in a name resolution process does not support such services.
- The *authoritive* parameter could be extended, or a *integrity_check* parameter added to the *fn_ctx_handle_from_initial*( ) and *fn_ctx_handle_from_ref*( ) functions to enable a caller to control the requirement for name resolution integrity checks for operations using the context handle returned.

  Note: The proposed extensions to DNS provide for data origin authentication and integrity. See reference **DNSSEC**.

5.2 Name Re-use and Re-assignment Policies

As previously indicated policy constraining the reuse or re-assignment of names may be imposed within some name services. Such a policy would have an impact on the specification of the *fn_ctx_bind*( ), *fn_attr_bind*( ), *fn_ctx_unbind*( ), *fn_ctx_rename*( ), *fn_ctx_create_subcontext*( ), *fn_attr_create_subcontext*( ) and *fn_ctx_destroy_subcontext*( ) functions.

A policy restricting the re-use or re-assignment of names may result in the functions performing bind or subcontext creation operations may fail because of a previous use of the name or because the re-assignment of the name is not permitted. This failure condition could be represented by an additional status code such as **FN_E_NO-REUSE**.

A name service implementing a name re-use or re-assignment policy could implement an unbind or destroy subcontext operation as a *hide binding or subcontext* operation. This changes the state of a binding so that it is ignored by normal name resolution requests. Facilities to manage such hidden bindings may need to be provided with the XFN API. Hidden bindings may need to be included in specific name resolution requests to satisfy historical audit analysis. Under specific authorisation hidden bindings may need to be deleted or archived. Such facilities may be supported by providing a separate set of functions, or including an additional parameter within existing functions, or using existing functions with a name service specific authorisation associated with the caller acting as a trigger for the specialised behaviour.

5.3 Audit of Name Service Operations

Some name service operations may be considered to be potentially security relevant and as such should result in the generation of an audit event that is submitted to a security audit service. All bind and attribute management functions would be included in this category. These would normally be audited by the name server servicing the request. It is a discussion point as to whether the XFN specification should require an XFN client to also audit such requests and their outcome.

Security relevant events within an XFN client that should require auditing are name resolution failures because of integrity failures and non-availability of service.

Glossary

access control
The prevention of unauthorized use of a resource including the prevention of use of a resource in an unauthorized manner [ISO 7498-2:1989].

access control policy
The set of rules that define the conditions under which an access may take place [IS0/IEC CD 10181-3 Oct 1991].

accountability
The property that ensures that the actions of an entity may be traced to that entity [ISO 7498-2:1989].

atomic name
Indivisible component of a name.

authenticated identity
An identity of a principal that has been assured through authentication [ISO/IEC DIS 10181-2 Jul 1991].

authentication
See data origin authentication, and peer entity authentication [ISO 7498-2:1989].

authentication exchange
A sequence of one or more transfers of exchange authentication information (AI) for the purposes of performing an authentication [ISO/IEC DIS 10181-2 Jul 1991].

availability
The property of being accessible and usable upon demand by an authorised entity [ISO 7498-2:1989].

binding
association of an atomic name with an object reference.

composite name
name that spans multiple naming systems. An ordered list of one or more components.

composite name resolution
the process of resolving a name that spans multiple naming systems.

compound name
sequence of one or more atomic names composed according to a naming convention.

confidentiality
The property that information is not made available or disclosed to unauthorised individuals, entities, or processes [ISO 7498-2:1989].

context
an object whose state is a set of bindings with distinct atomic names. Every context has an associated naming convention. A context provides a lookup (resolution) operation, which returns the reference bound to an object, and may provide operations such as for binding names, unbinding names, and listing bound names.

credentials
Data that is transferred to establish the claimed identity of an entity [ISO 7498-2:1989].

cryptography
The discipline that embodies principles, means, and the methods for the transformation of data in order to hide its information content, prevent its undetected modification and/or prevent its unauthorised use.

Note: The choice of cryptography mechanism determines the methods used in encipherment and decipherment. An attack on a cryptographic principle, means or method is *cryptanalysis*.

data integrity
The property that data has not been altered or destroyed in an unauthorised manner [ISO 7498-2:1989].

data origin authentication
The corroboration that the entity responsible for the creation of a set of data is the one claimed.

denial of service
The prevention of authorised access to resources or the delaying of time-critical operations[ISO 7498-2:1989].

digital signature
Data appended to, or a cryptographic transformation (see cryptography) of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery for example, by the recipientx[ISO 7498-2:1989].

DNSSEC
Domain Name System Security Extensions.

encipherment
The cryptographic transformation of data to produce ciphertext.

Note: Encipherment may be irreversible, in which case the corresponding decipherment process cannot feasibly be performed. Such encipherment may be called a one-way-function or cryptochecksum.

federated namespace
set of all possible names generated according to the policies that govern the relationships among member naming systems and their respective namespaces.

federated naming service
service offered by a federated naming system.

GSS-API
Generic Security Service Application Programming Interface. Independent Data Unit Protection.

initial context
every XFN name is interpreted relative to some context, and every XFN naming operation is performed on a context object. The XFN interface provides a function to allow a client to obtain an *''initial context''* object that provides the starting point for the resolution of composite names.

IPSEC
IP Security Protocol.

masquerade
The unauthorised pretence by an entity to be a different entity [ISO 7498-2:1989].

namespace
set of all names in a naming system.

naming service
the service offered by a naming system

naming system
a connected set of contexts of the same type (having the same naming convention) and providing the same set of operations with identical semantics.

naming system boundary
the point where the namespace under the control of one member of the federation ends, and where the namespace under the control of the next member of the federation begins.

peer-entity authentication
The corroboration that a peer entity in an association is the one claimed [ISO 7498-2:1989].

principal
An entity whose identity can be authenticated [ISO / IEC DIS 10181-2 Jul 1991].

reference
a reference of an object contains one or more communications endpoints (addresses).

repudiation
Denial by one of the entities involved in a communication of having participated in all or part of the communication [ISO 7498-2:1989].

secure association
An instance of secure communication (using communication in the broad sense of space and / or time) which makes use of a secure context.

secure context
The existence of the necessary information for the correct operation of the security mechanisms at the appropriate place and time.

security attribute
A security attribute is a piece of security information which is associated with an entity in a distributed system [ECMA-138 Dec 1989].

security audit
An independent review and examination of system records and operations in order to test for adequacy of system controls, to ensure compliance with established policy and operational procedures, to detect breaches in security and to recommend any indicated changes in control, policy and procedures [ISO 7498-2:1989].

security audit trail
Data collected and potentially used to facilitate a security audit [ISO 7498-2:1989].

security domain
A set of elements, a security policy, a security authority and a set of security-relevant operations in which the set of elements are subject to the security policy, administered by the security authority, for the specified activities [ISO / IEC CD 10181-1:Dec 1992].

security policy
The set of criteria for the provision of security services.

security service
A service which may be invoked directly or indirectly by functions within a system that ensures adequate security of the system or of data transfers between components of the system or with other systems.

SSL
Secure Socket Layer.

subcontext
an atomic name in one context object can be bound to a reference to another context object of the same type, called a subcontext, giving rise to a compound name. For example in */usr/local/bin* the atomic name *local* is bound in the context of *usr* to a directory context (and subcontext) in which *bin* is found.

threat
A potential violation of security [ISO 7498-2:1989].

traffic padding
The generation of spurious instances of communication, spurious data units or spurious data within data units [ISO 7498-2:1989].

vulnerability
Weakness in an information system or components (for example, system security procedures, hardware design, internal controls) that could be exploited to produce an information-related misfortune [FC Ver 1.0 Dec 1992].

Index