INTERNET-DRAFT
draft-hoare-nics-00.txt
expires July 15 1997

Graydon Hoare
<graydon@pobox.com>

Jan 15 1997

NICS
Network of Identifier and Credential Servers
(first public draft)

Status of this Memo

Abstract

NICS is a proposed system which meets the requirements of large-scale, unique principal identification, for use in conjunction with an arbitrary set of security systems such as have been proposed by members of the IETF.

This proposal outlines the motivation for the development of NICS, and gives a general description of its internal workings and interfaces with higher-level protocols.

It should be emphasized up front that NICS is not a complete security system, nor does it aim to replace any existing components of the internet which already work. The design draws off the fact that many security systems already have flexible name schemes, and are therefore considered components which are used in conjunction with NICS to achieve an improved level of service, flexibility and reliability, while introducing many desirable features such as anonymous identifiers, self-optimization, and low-overhead operation.

For the purpose of initial evaluation, the remainder of this paper is short and to the point, and requires a little work on the reader's side to understand the reasoning. Additional discussion is welcome on the mailing list.

Rationale

The intent of the author is to produce a stable, self-maintaining distributed database capable of identifying (naming) an extremely large number of principals and providing their credentials to any principals who ask. Such an intent initially appears to overshadow pre-existing systems such as DNS, X.500, SDSI, etc. However this is not the case: all currently existing naming schemes allow for, or require at some point in their operation a unique, unchanging reference name for a given principal. Currently existing standards to refer to each other by default: SDSI imports global names out of the DNS, X.500 imports names out of the MX map of the DNS, DNSSEC imports names from X.500/509 names, or from IP numbers etc. None of these exported names are static, or even acceptably detached from real-world organizational structures. Most standards also allow for additional naming schemes. NICS is an attempt to define such a naming scheme on which all others may optionally be built, or on which low-overhead security services may directly operate. NICS does not guarantee anonymity in all cases, nor does it guarantee any sort of directory service. The design does not specify a security framework -- the issue is implementation specific. The only feature NICS adds to the internet is strong universal naming, for purposes of customization, accounting, authentication, authorization, and other ``security'' procedures. It is beyond the scope of this initial document to describe these procedures further; the reader is assumed to be familliar with modern security systems.

Design

NICS is a system for serving the values of a single relating table to anyone connected to the global internet. The table consists of identifiers (which uniquely identify communicating principals) and their associated credentials. This is called the NICS Database. It is served by Identifier and Credential Servers (ICSs). The format for NICS IDentifiers (NIDs) is specific and fixed: they are all 16-bytenetwork order unsigned integers. The numeric space afforded by 16bytes is immense, as no NIDs go unused within NICS. In comprehensible terms, a 128-bit length would allow each of 6 billion people to allocate 1 billion new NIDs every milisecond of the day for 1.8 million millennia before the range is exhausted. It is therefore suggested that 16 bytes is sufficient even for the large task of unique identification.

The format for credentials is variable, but each type of credential must conform to the following criteria:

* it must be possible to encode, transmit and store the credentials as a string of bytes (octets)

* the standard from which the credentials are derived must be recognized and numbered by IANA.

Every principal within NICS is represented solely by its NID. A principal uses its NID and its associated credentials to establish any further relationships, but the basic NICS record is just a NID -> Credential mapping called a Principal Record (PR).

An ICS will use credentials within its local database to evaluate and authenticate transactions with other principals, including other ICSs. It is therefore essential that at minimum an ICS maintains its own PR in a safe, local store.

The database is distributed amongst all known ICSs in a non-hierarchical maner. It is devided amongst hosts by using Scopes. A scope is stored as a tuple of 1 low NID and 1 high NID, which together represent an inclusive numeric range. A scope is associated with the NIDs of one or more ICSs acting as replication-peers for that scope in a structure called a Service Record (SR). Every ICS in an SR's peer table is responsible for storing a copy of every allocated PR within the Scope. A change to a PR within a scope can be initiated by any SR-peer, and inter-peer synchronization is achieved through flooding of timestamped advisory locks and a simple commit/rollback scheme. Any peer in an SR is responsible for maintaining an accurate map of the SR, as well as an accurate map of the 2 SRs with ranges immediately ``above'' and ``below'' the SR. If a peer is a member of more than one SR (as will often be the case) it must maintain these records for each SR independently. For every change to a PR or SR, every peer in the SR-peer list needs to acknowledge the change. SR updates should additionally flood into the neighbouring SRs, although acknowledges from neighbours may be delayed or ignored.

Each SR has an implicit ``desired distribution'' which is calculated based on its breadth of scope, among other things. The desired distribution is the ideal number of hosts on which the scope should be replicated in order to be considered reliable and fast -- obviously the calculation will need to take into account a few factors. Every peer within an SR communicates the size of its remaining allocated local storage to every other peer. Expansion of an SR towards its desired distribution takes place using the standard ICS flooding transaction method. An ICS will introduce a new ICS into an SR that it is a member of, if and only if

* the desired distribution for the SR has not yet been met

* the new ICS has more free space available than any other willing neighbours

The first ICS within an SR to reach its highwater mark in allocated storage (which should be well beneath its real limit, as extra space is needed in panic situations) signals its peers that it is running out of space, and it is time for a ``split'' transaction. The SR then splits at a NID specified by the initiating peer, and a new SR is created beginning at the split NID + 1. The new SR excludes the initiating peer. This split takes place on all peers in the SR. The appropriate ``above'' and ``below'' SRs are adjusted, and the newly split scope goes about its top priority which is always to reach the desired distribution.

A peer may, at any time, ``resign'' from an SR provided the SR has other active peers. In the unlikely event that all peers in an SR resign or are disconnected in rapid succession, the neighbours of the SR will be made aware of the fact, and are obliged to offer assistance in transferring the portion of the database off the crashing servers.

Aggregation may take place using an as-of-yet undecided strategy. Since all peers within an SR are always aware of the NIDs of all neighbouring peers, it is not difficult to imagine a strategy of SR-boundary re-alignment in order to aggregate SRs. It should be stressed however that stability through replication takes priority over optimizing aggregations.

Any query which enters an ICS is answered by taking the following steps (assuming principal authentication)

* If the PR is stored locally, return the credentials of the query

* Otherwise, forward the query to a random member of the SR with the closest known range.

In these cases, forwarding can mean either returning the SR to the client or actually performing a recursive query, much like DNS. Usually a recursive query will yield better results. Anyone is allowed to cache SRs, so locating an appropriate server should not take too long. PRs are not intended to be cached unless they are flagged with a special loose-security bit, which enables cached PRs to use TTLs for cache-consistency. Any PR without the loose-security

bit set may be cached, but must have at minimum its hash-value reloaded from an authoritative peer every time it is required by another layer of the system. It will be hard to enforce this rule, but adherance is recommended in order to close any possible windows for attackers. In order to remain reasonably efficient in spite of the heavy consistency requirements, the communication protocol must be lightweight, and the cache of SRs much be as large as possible in clients.

For update messages, the query is processed in a timestamped, flooded two-phase commit, in which a transaction only proceeds after a signed cookie is received from all peers in the SR. If an SR peer is ever ``unreachable'', it is flagged as down and all commits are assumed to be approved by it. If the peer ever comes back up, in order to clear its down flag it must take whatever measures necessary in requested transfers to provide to all members of the SR with a signed digest of the most recent version of the SR. In the interim, the SR may have abandonned the peer altogether and expanded to the desired distribution using other peers. In such a case the host is informed of its exclusion upon resuming contact with its old SR. A peer is expected to abide by such a decision.

There is no defined means (yet) for dealing with multiple sets of peers who consider one another mutually exclusive. One SR must yield control of the given range to the other. Otherwise a ``shadow NICS'' emerges, much like the ``alternative'' domain name servers, except that no integration is possible because there is supposedly only one NID range. Such activity would therefore only be destructive to both sets of peers, as they would no longer be able to distinguish which NID was which. While such activity could be used as a means of attack, sufficient CA data communicated out of band makes this easily defended against.

Finally, announcement of an ICS proceeds exactly as does announcement of any other principal: the principal sends a broadcast challenge for valid authentication over its local link. The challenge uses either its own previously-established PR or an out of band PR. Any valid response it receives has proven itself to be attached to the wider NICS, and it therefore added to the client's list of local ICSs. If no response is received, a larger broadcast may be necessary, or the use of some service-location protocol. If an out of band PR is chosen, the next request will probably be a query to set a new NID for the principal. Such a query is forwarded a random number of times to random peers within the NICS and eventually serviced (perhaps after a TTL expires) by assigning the

next consecutive NID in a scope with free space. NIDs are assigned consecutively within scopes starting from a randomly chosen position and overflowing onto the low end of the scope if necessary (in order to better permit aggregation) but since scopes shift from one server to another the actual value of a NID does not reveal any practical information about the principal posessing it.

In fact, this is the primary motivation for NICS: principals may have as many unassociated and (to varying degrees) anonymous NIDs as they feel they need to maintain privacy. If a NID ever outlives its usefulness, its PR may be set to NULL at which point no further modifications are authorized to take place. It is ``dead''. It is the responsibility of the principal to keep joinable data out of

their PR, and out of the hands of anyone they do not wish to trace them, but under NICS it actually is an option to have anonymous identities.

Implementation

At this time, no implementation is available. The author has begun work on a portable ANSI C version of an ICS, but it will most likely need to be rewritten several times in order to conform to emerging GSSAPI & IPSEC standards. Most of the code required for such a server has already been written -- there are libraries to handle most authentication schemes, simple database management, caching, and network communication. It should be stressed that NICS is intended to be capable of operating in an extremely low-overhead system, such as a consumer electronics device or embedded controller. Several scenarios do not even include mutual authentication -- systems relying on smart-cards or even magnetic strip cards may benefit from a uniform identifier scheme.

In order to make NIDs easier to remember, certain transformations are suggested such as mapping 16-bit chunks into a dictionary of standard english words. A 2^16 entry dictionary was prepared as a demonstration.

The random NIDs (in 2^16 decimal chunks)

* 56184.10819.11346.28658.8732.65087.5944.14558
* 38012.49371.6317.16111.20713.58321.55011.48691
* 6887.1175.33979.52356.53123.62765.14518.24799

map to

* sucks.claw.coefficient.incredible.capioma.yountsville.bmw.dedeaux
* misery.rillton.bootstraps.dome.flandry.tick.squeaky.renshaw
* breathes.aliquid.liveware.shackleton.sides.ways.declez.gulph

which, while not exactly easy to remember, are better than their decimal counterparts. Such identifiers can be stored within any other security system easily as extended name types, revealing as little as is desired (or knowable) about the principal.

Further comments on NICS can be directed to the IPSEC mailing list or to the author at graydon@pobox.com.