Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 1, 2021

R. Van Rein
InternetWide.org

September 28, 2020

InternetWide Identities with Realm Crossover
draft-vanrein-internetwide-realm-crossover-00

Abstract

Domains and domain user identities are available in many protocols, and can be expressed as part of the URI grammar. This document outlines how clients can bring their self-controlled identities over when crossing over to foreign realms that rely on authenticated user identities.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 1, 2021.

Table of Contents

1. Introduction 2
2. Bring Your Own IDentity as a Usage Pattern 3
3. Grammar of Identities 4
4. Example Use Cases 5
4.1. Example of a Local Identity Grammar 5
4.2. Example Targets for Access Control 6
4.3. Example Regimen for Access Control 7
5. Realm Crossover Techniques 9
5.1. Realm Crossover for Kerberos 9
5.2. Realm Crossover for SASL 10
5.3. Realm Crossover for PKIX 12
6. New Application Protocols 13
6.1. Remote PKCS #11 13
6.2. Keyful Identity Protocol 13
6.3. Helm Access (from Arbitrary Nodes) 14
6.4. InternetWide Roaming 16
7. Normative References 17
Author's Address 19

1. Introduction

Many protocols identify clients and servers through a domain name or a user at a domain name. Domain names follow the stepwise delegation of authority that is engrained in DNS, and an added username is generally considered a further refinement that falls under the authority of the named domain.

URI grammar mirrors this idea in its authority section. Some additional information is present to facilitate resource location beyond an identity; these involve the scheme and an optional port, and for some schemes there may be a host name as a mild overspecification for a domain.

InternetWide Identity, as introduced herein, allows domain.name and user@domain.name identity forms across protocols, and when included in a URI it treats any path, query part, port, URI scheme and host-instead-of-a-domain as information beyond the abstraction level of interest to identity. In other words, variable paths, host names, ports and service protocols can occur in URIs that represent the same identity.

InternetWide Identities are domain-scoped and intended for use in foreign servers that may reside in the client's domain or any other domain. We informally refer to this idea as "Bring Your Own IDentity (BYOID)" and to the technology facilitating it as "Realm Crossover" for domain.name and user@domain.name identities.

Within a protocol, the client and service may each have their own identity and when represented as a URI they may differ in many ways, specifically including a possibly different domain name. This is common for communication protocols such as SMTP or XMPP. It is less common for protocols granting access to resources, like HTTP, where resources may be specified with a URI but client identity is an after-thought that gets mixed into that one URI as though it were a service-side identity. This conflation of client and service domain relaxes when the client identity is described with its own URI, allowing better integration with the client realm and other services working for it.

Foreign services generally implement some form of access control, founded on an authenticated client identity. The process of authentication validates the client domain name through such mechanisms as DNSSEC with DANE and TLS. An identity callback to the client realm can then add the user part of the client identity, according to a source whose prerogative it is to define it. The foreign service composes the domain.name that it validated for the client with this user part to find a user@domain.name.

Since the client's username is provided under a domain's authority over user names, it is possible to modify that part before it is supplied to the foreign service. The name of the foreign service can be a parameter in making this choice, especially when it is authenticated. This point of change to the user name can be helpful to change to an alias for client privacy; it may be used to slip into a group or role; and it can support clients from yet another realm to be represented by an alias or group or role under the client domain.

This document only outlines the ideas and protocol modifications that can realise them. Specifics for each of the implements are deferred to separate documents, even when the concepts described herein have already been shown to work in code.

2. Bring Your Own IDentity as a Usage Pattern

The general usage pattern introduced with InternetWide Identity is one where a client controls a set of usernames, residing under a realm of its own choosing. This client realm is implemented under a client domain name. The client may approach services running under the same or any foreign domain. In either case, the client brings their own identity composed of the client username and client domain; the client realm actively facilitates authentication under this composite identity.

As part of client control over their own identity, a service-specific client username may be selected from among a set of pseudonyms

available to the cient. This enables the client to manage their identity, and the client realm can provide a number of forms to facilitate this; clients may create fresh identities or offer "+alias" extensions, or switch to any of a group member or role occupant, or even to a shared identity for an entire group or role. The client realm or their user agent may remember choices made in the past and suggest them again during new encounters with the same service.

The design challenge of InternetWide Identity is to facilitate these patterns in current-day protocols. This calls for additional Realm Crossover protocols and techniques, and the sections below outline how application protocols with Kerberos [Section 5.1] and/or SASL [Section 5.2] authentication can be extended to connect client and service realms, usually without modifications to application-layer protocols. It also explains how a distributed Public Key Infrastructure can be relied upon with similar techniques.

The general pattern of Realm Crossover is founded on the two-level authority of a user@domain.name identity. Though host, port and protocol as well as path and query string may be useful to locate a resource and for that reason incoporated into a URI, the proposed InternetWide Identity abstracts from those elements to allow shared identities across a variation of services and protocols. Only the domain and the username are considered for identity. The foreign service starts by authenticating the domain and makes an identity provider callback secured with mechanisms like DNSSEC, DANE and TLS. The callback should be validated to a point where its authority over usernames under the domain is certain. The callback can then be used, in a manner specific to the Realm Crossover technology, to authenticate the username underneath its domain. The composition of these two elements, username and domain, with an "@" to separate the fields, forms the full identity as it is further considered by the foreign service. This approach can be used for client identities, but may even be useful to validate service identites.

3. Grammar of Identities

The grammar of client and service identities are related to the definition of a NAI with UTF-8 support [RFC7542]. However, the NAI defaults to realm-internal use but BYOID always needs to express the domain, so the root of the grammar tree is different:

```
identity =  utf8-username "@" utf8-realm
identity =/                   utf8-realm
```

The grammars of utf8-username and utf8-realm follow the NAI specification [Section 2.2 of [RFC7542]] and neither may include the

"@" character. The incorporated UTF-8 grammar [RFC3629] only allows the shortest representation for each code point.

The utf8-username and utf8-realm are both considered to be case-insensitive, so technologies relying on Realm Crossover techniques can infer identity equality when nothing differs but for letter case.

The utf8-realm is a domain name under which users may be defined; note how this domain is not represented in the Internationalised Domain Name form (IDNA) [RFC5890] that is used on the wire in DNS, but as an utf8-realm that can be mapped from or to IDNA for DNS-related purposes. This allows rendering of domain names to users in the international form that they expect.

Host names, if they occur in a URI or as part of a protocol, must be translated to a domain name in a manner that may be specific to the protocol, but a reasonable general strategy might be to allow precisely one level to be stripped off when no definitions in DNS suggest otherwise.

For consistent BYOID portability, length considerations MUST NOT be constrained by support infrastructure beyond a general minimum, which follows the email limits [Section 4.5.3.1 of [RFC5321]] and IDNA. The utf8-username MUST be supported with sizes up to 64 octets and the domain in its DNS wire form MUST support sizes up to 255 octets. Note that the DNS wire form can be more compressed than the utf8-realm [Section 4.2 of [RFC5890]] because UTF-8 can use up to 4 octets for a Unicode code point, while IDNA can get up to a full Unicode code point per DNS octet once it is initialised; a safe minimum size to be certain to hold any utf8-realm is 1020 octets. The size of an identity MUST therefore be supported with sizes up to 1085 octets. Note that this imposes constraints on usable RADIUS implementations and the RADIUS User-Name attribute cannot be used to contain the utf8-realm value; the User-Name in a RADIUS implementation may even be unsuitable to carry the utf8-username, which for BYOID portability MUST be supported up to 64 octet sizes, whereas RADIUS permits a maximum size of 63 octets in the User-Name field.

4. Example Use Cases

This section informally describes use cases that may be useful.

4.1. Example of a Local Identity Grammar

Under the prerogative of a realm's identity provider a more specific grammar may be used, and some forms may even be distinguished by notation. Such distinctions however, MUST NOT be assumed by foreign

services and SHOULD NOT be assumed by anything in the provider's realm if that would break BYOID portability. Having said that, we do give a few informative ideas:

- john@example.com might be a user
- john+cook@example.com might be a light-weight alias, administratively scoped under user john@example.com
- mary@example.com might be a different user, or an external user granted a local alter ego, or an unrecognisable pseudonym for john@example.com
- cooks@example.com might be a group of users that can be addressed all at once by a variety of services, each with their own group semantics
- cooks+john@example.com might be a group member for cooks@example.com, as an in-group pseudonym for a user under the same or another domain
- +stove@example.com might represent a service
- +stove+nr3+elt5@example.com might represent a service with sub-addressing parameters nr3 and elt5

4.2. Example Targets for Access Control

The targets of access control are often called "resource". We specify a few forms that may be useful to consider, without it meaning to be the last word.

One form worth considering might be a Communication ACL. For a given local identity, this ACL would list the remote (client) identities that are welcomed for communication. A concrete usage pattern of this mechanism could be to blacklist everything but a few welcomed peers; or to whitelist everything but for invasive parties; or to graylist access and require the prospective communication partner to jump through some hoops before being welcomed. It is RECOMMENDED to verify access from a remote identity before initiating communication with them, thus ensuring their ability to respond to the sending identity.

The form of a Communication ACL is very general, and covers any protocol that connects a remote identity with the local identity in control of the ACL. So, after sending an email over SMTP, a response over SIP is also possible and LDAP may grant download for public keys for the local identity based on the same Communication ACL for the

local identity. But while the Communication ACL spans across protocols, it should not span identities; a local user is likely to use pseudonyms for the specific purpose of regulating access to this identity. Crossover should at best be limited to redirection from an access-blocking identity to a welcoming one.

Another general form could be that of a Resource ACL, which may be centered around an identity in the form of a UUID [RFC4122]. Again, this need not be constrained to a protocol but may represent more general ideas, such as "storage space" or "versioning system". Whether these are delivered over HTTP, LDAP, MQTT or GIT should not matter to the access control mechanism. Related, but different, are Resource Instances, which expand a general idea with a UUID-specific string format to describe a particular instantiation of the general idea; for example, a collection or object in a storage space; for example, a repository or version in a versioning system.

4.3. Example Regimen for Access Control

Access control is exercised while a service is accessed. The process starts with an authenticated client identity and derives what access rights may be granted to the identified client.

The facilitation of BYOID in access control demands that the realm in the client identity is fully taken into account. When any local realm is at best used for compressed representation but not to differentiate rights, an access control solution can be completely open to foreign clients.

One difference between foreign clients and local ones is the level of trust bestowed on them, another is the level of knowledge about their identity grammar. Specifically, the username forms suggested above for "+service" or perhaps "+service+arg1+arg2+arg3", for "client+alias" or "groupname+membername" may all have local meaning, especially because for a given name part "service" or "client" or "groupname" there is some local knowledge about the kind of identity that they represent. Though being permissive to foreign clients is vital for BYOID, differentiating rights for local identities (and identity patterns) can be useful in many operational contexts.

As a general mechanism for access control, it is RECOMMENDED to iterate over a client identity by gradually generalising the form. Each of the forms might be called a "selector". Going from concrete to abstract, ach Selector could serve as a (partial) identity string while looking for matching access rules. Such lookups could also incorporate the target of access control. Flags representing specific access rights such as read and write could be part of the lookup or returned as a result of it. The first identity for which

an access rule matches is considered the proper one for the client identity being iterated over.

The most specific username is the utf8-username field that passed authentication. The most abstract username is a wildcard, which we shall write as "". For local users, it may make sense to iterate over more specific forms as well, such as splitting after internal "+" signs. For example, "john+cook" could yield a list "john+cook", "john+" and "" going from specific to general.

The most specific domain name is the utf8-realm field that passed authentication. The most abstract domain is the top-level domain ".", and intermediate domains can be written with a prefixed dot to distinguish them from a full domain name. For example, "example.com" could yield a list "example.com", ".com" and "." going from specific to general.

The combined iteration for a client identity uses domain iteration as an outer loop and username iteration as an inner loop. So, combining the examples above including local interpretation of the username part, "john+cook@example.com" could yield a list "john+cook@example.com", "john+@example.com", "@example.com", "john+cook@.com", "john+@.com", "@.com", "john+cook@.", "john+@.", "@." where the latter is always the most abstract form found for a client@domain.name form.

Most of this example list is likely to remain unused; several of these forms could be blocked with pragmatic rules, such as barring ".com" domains from an ACL. When also the local interpretation of the username is barred, the lookups for such a remote client are reduced to "john+cook@example.com", "@example.com", "john+cook@." and "@." forms only, but for "john+cook@labs.example.com" there would still be the meaningful forms "john+cook@labs.example.com", "@labs.example.com", "john+cook@.example.com", "@.example.com", "john+cook@." and "@." to consider in the ACL.

Note the importance of the initial dot in the domain name iteration. It differentiates identities as defined by a realm from identities as defined by a child realm. One other choice that an ACL might make is to constain the iteration to go up one domain level and allow "." as a wildcard domain.

In general, the application calling for access control is in the best position to determine what to consider. An SMTP port 25 for external submissions may not want to consider internal identity forms, whereas an SMTP port 587 for submission by internal users may desire just that.

5. Realm Crossover Techniques

This section details how Realm Crossover may be established with Kerberos and SASL, and how PKIX may benefit from it. The descriptions are provided to sketch the structure of the solution, but complete details are worked out in other documents.

5.1. Realm Crossover for Kerberos

Kerberos uses a realm controller known as the KDC. It is possible for the KDCs of two realms to connect, though this is not a dynamic process that can be called upon when a client first attempts to acces a foreign service. Realm Crossover for Kerberos is an "Impromptu" extension to the existing facilities for Realm Crossover.

Only the client KDC and service KDC need to support Realm Crossover to allow this to work. Most current-day clients facilitate principal name canonicalization [RFC6806], which suffices for Realm Crossover. The client KDC needs to realise that a request was made for a service that is part of an external realm, obtain a (new or pre-existing) realm-crossing relation with that realm's KDC, and respond with a server referral [Section 8 of [RFC6806]] in order to redirect the client, armed with realm-crossing credentials, to the KDC of the service realm.

For the creation of a new, impromptu realm crossover relation between two KDCs, a new protocol KXOVER has been devised. It builds on TLS and derives a crossover key from the TLS master key. As part of the KXOVER protocol, details such as a time window for crossover are negotiated. Since the same crossover key can help any client on one KDC to connect to any service on another KDC, this procedure is efficient. Only the first "impromptu" call upon the remote KDC can be somewhat time-consuming; beyond that, the procedures fall back to symmetric key management only. When a crossover ticket is used regularly, it would make sense to refresh it before the old ticket expires.

It is worth noting that the secrecy of the crossover keys established over TLS are only as secure as TLS is. When no Post Quantum mechanisms are used, the crossover key derived by the client and service realm could be grounds for future abuse. The security model of Kerberos permits derivation of all derived keys and that includes encryption keys. Although Kerberos with proper initiation can be a Post Quantum technology, this may no longer hold under Realm Crossover.

Kerberos supports anonymity, but that would not allow distinction between different clients. It does not facilitate pseudonyms when

requesting a particular service on another realm; the KDC however, is in a perfect position to do just that. Extensions can be easily imagined, whereby a client sets a flag in the initial exchange to indicate support for client identity changes, either from the KDC's recollection of previous desires or from an explicit request by the client. As long as the KDC reflects this flag, identity changes could be requested or allowed as a protocol extension.

The current DNS records for Kerberos specify where to find the KDC for a realm, but not for a service. This was initially considered insecure, but since then DNSSEC has been rolled out, and it can now be done securely. This means that a client or its KDC can look for a service host or domain, infer the realm under which it resides, and lookup the ticket for the protocol and host/domain under that realm. This is not current practice yet.

In summary, the extensions to support Kerberos Realm Crossover are (1) the KXOVER protocol, (2) DNS lookup of realm names for hosts/ domains, (3) optional support of pseudonymity. With the exception of the optional last point, these can be implemented in the KDC alone.

5.2. Realm Crossover for SASL

SASL authentication is built into many Internet protocols. It facilitates an extensible set of mechanisms, passed inside the application protocol which remains blissfully unaware of its details and upgrades.

It is not generally safe to carry SASL over plaintext connections, and the customary use case runs it within a secure application protocol, such as after a STARTTLS exchange. To avoid undetected relaying of SASL traffic to another resource, a precaution of channel binding may be used, where non-secret but unmistaken parts of the secure application protocol are mixed into a cryptographic computation to assure that the SASL server is the assumed one.

To support Realm Crossover, it is possible to pass SASL to a backend server over Diameter [RFC6733]. The backend would be selected for the client's realm and looked up with Diameter's NAPTR and) SRV records in DNSSEC and connected under protection of DANE and (D)TLS. This is effectively an identity callback to a client realm. Under Diameter, authentication is bidirectional, so the client realm is aware of the requesting service realm.

Diameter is valuable because it is designed to authenticate across administrative domains, and pass success or failure along with descriptive attributes such as an authorisation username to a requesting service realm. This usage is called Network Access

Service [RFC7155]. The value of passing just success or failure by default is that no resources are made available; as a result, a Diameter service can be made available as a public authentication callback service, unlike a resource-supplying protocol such as IMAP.

Care must be taken to not ever pass traffic back up from Diameter to a higher-level protocol. SXOVER ensures this by loosing vital information in that case, namely for channel binding. Services other than Diameter MUST reject externally supplied channel binding octet strings and instead form their own. That way, a resource-supplying application protocol could not possible be attacked in the backend of a foreign service contacted by the client. That is, not when channel binding is being used. The flip-side of this coin is that the foreign service contacted by the client MUST relay channel binding information to its Diameter backend, which then forwards it to the point where cryptographic computations are performed, in the client realm. Channel binding effectively becomes the client's authentication of the foreign service.

Based on this, only SASL mechanisms that support channel binding are suited for Realm Crossover. There is another requirement however, and that is end-to-end encryption. Given that SASL mechanisms are not generally safe to pass over unencrypted channels, they cannot generally be trusted to pass through a foreign service either. A secret between the client and its realm must be obtained beforehand, and used to encrypt the exchange. Note the relative ease of obtaining such a key relative to proving client identity. A special SASL mechanism can then employ this secret, along with channel binding information, to securely wrap another SASL mechanism that does not need to live up to these requirements. That special SASL mechanism is GS2-SXOVER-PLUS; it is possible that other mechanisms are devised to allow the same usage pattern however.

Two last concerns to note for the special SASL mechanism are that it must mention the client's realm in a form readable to the foreign server. This part will also be validated by the foreign server, while making the Diameter back-call. The wrapped SASL mechanism therefore does not supply the client realm, but only its username, precisely its prerogative to define. Also, the form of channel binding used must be visible to the foreign server, so it can pack the information for relaying over Diameter. This is generally possible with GS2 mechanisms, though other forms can have specialised representations for the same information.

Note that not all these concerns necessarily apply for clients that are local to the "foreign" service. Even when they may also pass over Diameter, their traffic may remain "internal" and "trusted" and therefore more mechanisms may be available than for a truly foreign

client. It is common in SASL to allow the server to present acceptable SASL mechanisms, so this can be part of operational practices.

To allow SASL over Diameter, a few attribute-value pairs need to be defined under its Network Access Profile [RFC7155]. These are (1) SASL-Token, for relaying SASL's binary octets of a token in either direction, but only when one is supplied by the SASL endpoint; (2) SASL-Mechanism, a string listing space-separated mechanism names that are acceptable to the foreign service, or selected by the client; (3) SASL-Channel-Binding, to relay binary channel binding information from the foreign service to the client's Diameter server, and to allow the client and its realm to work out that no extra resources are in the loop.

Not all application protocols support SASL. Modern IRC does, and may benefit from incorporating Realm Crossover to make abusive patterns less likely, and to reserve usernames for returning clients. The most notable omission is HTTP, which resorts to higher layers, often involving manual actions and code mixtures with executable content from uncontrolled remote sites. An extension of HTTP with SASL is part of our proposal for realm crossover as an overall solution.

In summary, the extensions to support SASL Realm Crossover are (1) support for SASL over Diameter, (2) the GS2-SXOVER-PLUS mechanism and its incorporation into user agents and SASL over Diameter; (3) the reliance on Diameter for client authentication in foreign servers. Finally, (4) non-SASL-aware protocols need to be extended to support SASL.

5.3. Realm Crossover for PKIX

The definition of PKIX references LDAP for the retrieval of certificates from a certificate's DistinguishedName. Given the potential for access control when Realm Crossover enables requesters to authenticate, arbitrary privacy controls can be enforced using these mechanisms. LDAP services can pass through STARTTLS and the certificate used may be assured through DNSSEC and DANE to allow remotes to validate the authority of the information in LDAP, specifically when its DistinguishedName patterns coincide with the domain name [RFC2247] and when usernames are located with (uid=...) search filters.

When identities are managed with the BYOID intent, it makes sense to create key pairs for PKIX, OpenPGP and perhaps OpenSSH and store those in LDAP for retrieval by approved parties. This establishes the long-missed Public Key Infrastructure for the Internet, to benefit security of email and most other forms of communication.

Note how PKIX currently relies on "central" organisations to approve of certifictes, and how LDAP may be supportive of a more distributed mechanism. Also note how the mechanisms can be combined for maximum strength; one example use of this would be to facilitate federations with authentication that spans the Internet plus a root key that facilitates just the federation.

6. New Application Protocols

The idea of BYOID and its technological embedding as Realm Crossover allow a few useful ideas to be worked out in new application protocols. The general idea of these protocols is described below, but the complete details are described in other documents.

6.1. Remote PKCS #11

Clients may want to use key-based mechanisms even on platforms that cannot be trusted to protect these well, such as a mobile phone or other easily lost device. Such applications, as well as the desire to use the same keying material on multiple devices and the reflection that operational control of private keys with proper rollovers is difficult and may for a large portion of users better be left to their administrators, all combine to the idea that a Remote PKCS #11 service may be useful.

When identities are generated along with public key certificates that can be looked up in LDAP, it is useful to facilitate the matching private keys in such a Remote PKCS #11 service.

We have prepared an LDAP scheme to reflect private key references alongside public key certificates (with intended readability only to the key owners) and we have prepared a direct mapping of the PKCS #11 interface to a request/response format that supports locally callable functions that are actually applied remotely to private keys contained there, perhaps in a highly secured context.

6.2. Keyful Identity Protocol

The enhancement of authentication with Realm Crossover allows a great diversity of new use cases; but encryption still relies on the publication of public key certificates by recipients. When the requirement to encrypt sensitive content exists however, there is an immediate urge to have all recipient's public keys and not make an exception by sending unencrypted content to one recipient that lacks one. In short, there is a desire for sender-initiated encryption.

We devised a Keyful Identity Protocol (KIP) to fulfil this need. Its essential function is to take in a symmetric key and wrap it in a

manner that can only be decrypted by an authenticated party. In other words, it provides encryption to any client who can authenticate. The actual content is not passed over KIP, but the session keys are.

Encryption of session keys is done with a fixed key stored in the KIP service, and cryptographically bound to an access control list. Requests to authenticate require no authentication, and are ideally performed on a KIP service running under each recipient's domain. To request decryption, a recipient must authenticate, and their session key can be provided when the cryptographically bound access control list allows it.

KIP can also be used to sign a checksum, which is a notably different service from authentication. This requires authentication by the signing party and is best done at their realm's KIP service. To gain trust, a recipient would contact the KIP service under a (validated) signer's realm and see it approved.

KIP Documents are a nested sequence of data and wrapped session keys and signatures. This involves signed references, so an object may be stored outside of a KIP Document, and still be validated by it. By defining a media type for KIP Documents, a data URI [RFC2397] can be made that triggers a handler when accessed by user agents such as HTTP and SMTP clients.

The wrapped session keys of KIP underly the GS2-SXOVER-PLUS mechanism Section 5.2, without other dependency on the protocol; KIP service can be used to help clients to the initial key that provides end-to-end encryption between their user agent and the identity callback server in the client realm.

In spite of its potential strength, KIP is operationally really simple to deploy. It requires a secure storage mechanism for its fixed long-term key, but the only administrative control needed is the creation and destruction of virtual hosts; and a basic roll-out only needs one. KIP can be a client to Realm Crossover for straightforward authentication.

6.3. Helm Access (from Arbitrary Nodes)

Domains may seem to be the anchor point for online identity, but in reality they often have fallbacks to a single email address, to which unencrypted email is sent. Other online resources reproduce this insecure habit. Email address may go awry, and with it, possibly the control over a domain name. The general problem here is that no problem exists for "identity bootstrapping". Realm crossover can be used to remedy that.

Online resources usually have a starting point, where administrative control may be exercised, including the removal of the resource and creation of new ones. It may even include the addition of subordinate identities, such as domain users or subdomains under a domain name. We introduce the general idea of a "Helm of Control", or helm for short, as the point of access to this facility.

We propose the use of Realm Crossover to gain access to one's helm. This involves authentication with a (possibly external) credential and not the ability to receive unencrypted email, so it is already more constrained. In addition, it should be straightforward to control in a more refined access how control can be held. At the very least, it is desirable to allow access form multiple (external) identities to have no single point of failure when it comes to access recovery.

Use cases may involve depositing credentials with others who can serve as fallbacks for access. This is especially useful by handing them their own access control, possibly leading to their own Helm, where they can only control those online resources that they are supposed to control. In general, a many-to-many mapping from identity to online resource at a given provider seems the most useful.

One use of this access pattern include control over the removal of a person's online presence, for instance by family after a loved one passes away or after a company goes into bankruptcy. At least some laws discard an individual's right to privacy upon death and may therefore not be supportive of a family's desire to erase online presence without going through disturbing practices in full control of a service provider; and not all of them see a benefit in destroying data.

HAAN is short for Helm Arbitrary Access Node; the HAAN service generates a worthless username and password from scratch. Note that any fresh account is without value until it is setup with online resources, perhaps by accessing a Helm. Though supplied by a given realm, the HAAN service for a realm does not even need to store the credentials; it generates a random username with a lot of entropy and uses an internally held, fixed key to derive an accompanying password. Later, when a client wants to gain access, the provided username is combined with the fixed key to recompute the password and perform any desired check.

The implementation of HAAN is founded on a SASL mechanism named GS2-HAAN-GENERATE, which should be run directly with the supporting realm (or perhaps through an end-to-end encryption tunnel like the one for SXOVER). Later authentication is possible using any SASL mechanism

that uses a password, usually with GS2-SXOVER-PLUS wrapped around it so a foreign provider can use it to gain access to its Helm.

It is not generally advised to require authentication when adding a HAAN identity to a Helm. First, it exchanges more credential information than strictly desired; second, users do not type identities from slips of paper but use copy/paste; third, backups exist in the form of additionally registered identities to access the helm; fourth, because HAAN can generate extra entropy to allow correction of typo's when a client enters a falsely entered identity as an authorisation identity. When desiring a 128-bit security level, HAAN might generate 160 bits and as long as going from the proper 160-bit value to an authorisation identity can be described with entropy less than the extra 32 bits, the validation with HAAN can still provide the 128-bit security level.

Dedicated methods can be built for HAAN-generated secrets. One that springs to mind is TOTP [RFC6238], where the only storage needed on the server is for replay avoidance, and only when nothing specific to the SASL exchange itself avoids replay. When replay is not to be suspected, the TOTP exchange can be accepted immediately; otherwise, a challenge/response exchange may remedy any risk of replay. The server can safely err on the cautious side, and use something like a Bloom filter to detect possibly replay over the past timer period.

In spite of its promising potential, HAAN is operationally really simple to deploy. It requires a secure storage mechanism for its fixed long-term key, but the only administrative control needed is the creation and destruction of virtual hosts; and most roll-outs will only support one.

6.4. InternetWide Roaming

Networking is increasingly becoming an application under management of users. Network authentication is customarily generalised in EAP, whereas application authentication is generalised with SASL. An ability to run SASL over EAP supports the use of client credentials for network access applications such as VPN, 802.1x or Bluetooth.

When SASL is backported over Diameter, an extra option arises, where the user is offered a tunnel to a home network. Using ESP, such a tunnel would be protected from content inspection and manipulation. And quite interestingly, ESP could be carried over a multitude of protocols.

When providing network access, this scheme has the benefit that nothing is known about the traffic, and so no responsibility is taken for those contents. Furthermore, the IP address from which this

content is shared is only a tunnel endpoint to a home realm, but not a general carrier for traffic. This could be a type of service to offer without concerns.

Packets of an ESP tunnel may travel over a variety of carriers, not just over IP. There could be an embedding for directly carrying it over ethernet or PPP, for example. There is an additional need for key exchange, but the UDP encapsulation for ESP [RFC3948] solves that with a simple prefix to the key exchange traffic, so that a complete tunnel protocol arises. This tunnel protocol can pass through PNAT using the UDP encapsulation.

This allows wired and wireless ethernets to employ 802.1x access restriction with SASL over EAP, to open up an ESP/IKE uplink to a tunnel server whose address information is provided in a Diameter response when SASL over Diameter succeeds. A base station may respond to a query for an "InternetWide Roaming" identity with just this dialog.

Similar approaches might be employed with the Bluetooth Network Encapsulation Profile to grant nearby nodes access to an ethernet, albeit at much lower speed, but can reduce the ethernet header to just the ethernet type. As a fallback scenario, mobile users may be helped with the older L2TP/IPsec stack which is available by default in most current-day configurations and this may serve as a fallback in circumstances where InternetWide Roaming is not available in a more direct form, but generic Internet access is. The simplified access form however, can be particularly interesting for small devices, especially when the ESP traffic is founded on a fixed key.

The integration of SASL into EAP adds a use case for key extraction from SASL, which is sometimes considered an older possibility, but which regains interest when it can setup end-to-end protection keys. In a scenario where SASL is followed by ESP, it may provide extra entropy to be mixed into the key exchange [RFC8784].

7. Normative References

[RFC2247] Kille, S., Wahl, M., Grimstad, A., Huber, R., and S. Sataluri, "Using Domains in LDAP/X.500 Distinguished Names", RFC 2247, DOI 10.17487/RFC2247, January 1998, <https://www.rfc-editor.org/info/rfc2247>.

[RFC2397] Masinter, L., "The "data" URL scheme", RFC 2397, DOI 10.17487/RFC2397, August 1998, <https://www.rfc-editor.org/info/rfc2397>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, DOI 10.17487/RFC3948, January 2005, <https://www.rfc-editor.org/info/rfc3948>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <https://www.rfc-editor.org/info/rfc5321>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.

[RFC6238] M'Raihi, D., Machani, S., Pei, M., and J. Rydell, "TOTP: Time-Based One-Time Password Algorithm", RFC 6238, DOI 10.17487/RFC6238, May 2011, <https://www.rfc-editor.org/info/rfc6238>.

[RFC6733] Fajardo, V., Ed., Arkko, J., Loughney, J., and G. Zorn, Ed., "Diameter Base Protocol", RFC 6733, DOI 10.17487/RFC6733, October 2012, <https://www.rfc-editor.org/info/rfc6733>.

[RFC6806] Hartman, S., Ed., Raeburn, K., and L. Zhu, "Kerberos Principal Name Canonicalization and Cross-Realm Referrals", RFC 6806, DOI 10.17487/RFC6806, November 2012, <https://www.rfc-editor.org/info/rfc6806>.

[RFC7155] Zorn, G., Ed., "Diameter Network Access Server Application", RFC 7155, DOI 10.17487/RFC7155, April 2014, <https://www.rfc-editor.org/info/rfc7155>.

[RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, DOI 10.17487/RFC7542, May 2015, <https://www.rfc-editor.org/info/rfc7542>.

[RFC8784] Fluhrer, S., Kampanakis, P., McGrew, D., and V. Smyslov, "Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security", RFC 8784, DOI 10.17487/RFC8784, June 2020, <https://www.rfc-editor.org/info/rfc8784>.

Author's Address

Rick van Rein
InternetWide.org
Haarlebrink 5
Enschede, Overijssel 7544 WP
The Netherlands

Email: rick@openfortress.nl