InternetWide Identity with Realm Crossover

OpenFortress*
digital signatures

seasons of change

Winter for percolation and reflection. . .
* Bring Your Own IDentity
* Identity, Access Control, Groups

Spring for support libraries, demonstrations
* ARPA2 Common: Identity, Access, Group
* Realm Crossover: SASL, Kerberos

Summer for protocols and applications
* Apache, GnuTLS-KDH, Postfix, Reservoir, KIP, HAAN

Autumn for harvesting
* IETF standardisation
* Domain Hosting provisioners
* Users gaining control over their online presence

thanks for support

Many have supported our work:

* NLnet	liberally supports this project
* SIDNfonds	supports our work on email with ARPA2 Identity
* SURFnet	supported KXOVER
* RVO	has supported our developers through WBSO
* NGI Pointer	from the EU intends to support Realm Crossover

Thank you all for making this possible!

current activity

Henri Manson is our integrator.
Works on SASL over Diameter, Apache modules.

Adriaan de Groot is our build hero.
has a focus on LDAP, and on project/build infrastructure.

Tom Vrancken is our 2nd cryptographer.
Works on TLS-KDH, including the GnuTLS code.

Rick van Rein is the project architect.
Works on specifications / IETF work, cryptography, API design,
fitting puzzle pieces together.

We do this work strictly for open source / protocols / standards.

steep challenge

We want to make an identity system for the Internet
* Grant users full control over their online identity
* Support them with easy-enough security and privacy

We do not take this lightly. . .
* This is not just for the web
* This is not just for one domain
* We retrofit the ideas into existing protocols
* We retrofit the ideas into existing software
* We take the effort to write Internet Drafts
* We take the effort to discuss Internet Drafts

The big question is usually
* Can we map our model to others?

bring your own identity

Controling your online identity?
Bring it under your own domain name

john@example.com

* Domains like example.com can be validated
* Identities like john are distributed by the domain
* User naming is the prerogative of a domain

* Now we need protocols to do this. . .

  → Most protocols incorporate SASL or Kerberos
  → Others can usually add SASL (HTTP, EAP)

See: draft-vanrein-internetwide-realm-crossover

realm crossover for kerberos

Kerberos identities are the prerogative of a KDC

KDCs can crossover via a special ticket,

krbtgt/EXAMPLE.ORG@EXAMPLE.COM

* This is issued to clients of example.com
* It points to services under example.org
* Kerberos is funny about capitalisation

KDC crossover involves (manual) key exchange
* We automate this with DNSSEC/DANE/TLS
* KXOVER = Realm Crossover for Kerberos
* https://gitlab.com/arpa2/kxover

realm crossover for sasl

SXOVER = SASL with end-to-end encryption wrapper
* Client shares a key with its own IdP
* SASL flows through a service back to the IdP

Need to make callbacks from any application
* Adding SASL attributes to Diameter
* Diameter is like RADIUS, but for untrusted networks
* domain SRV $\rightarrow$ SCTP/DTLS peering $\rightarrow$ secure traffic

Applications connect to a local Diameter agent
* Using a simple library and a TCP connection
* Applications may fork, thread, event-loop

Very easy to add to applications
* We provide an Apache module, KIP, Reservoir, . . .
* Application programmers should feel free to pick it up

realm crossover for certificates

uid=john,dc=example,dc=com

* DNSSEC can assure the dc=example,dc=com part
* A domain CA could validate the uid=john part
* DANE could confirm the validity of the domain CA
* _client-identity.example.com IN TLSA . . .

* *Currently just an idea*

Many use cases:
* Client identity for S/MIME (encryption, signing)
* Client identity for TLS protocols (login)

arpa2 identity

Identities are *Network Access Identifiers*
* RFC 7542 form utf8-username "@" utf8-realm
* We delegate the complexities of Punycode to DNS

The utf8-username can have + between words:
* john is a regular user identity
* john+cook and john+cook+vegan are aliases
* +docs+1234 is a service docs with argument 1234
* john+G9ASORZGC3DBNRQQU+ is a signed identity

http://common.arpa2.net/md_doc_IDENTITY.html

arpa2 signed identity

john+G9ASORZGC3DBNRQQU+@example.com is a signed identity

* Aimed at user john@example.com
* With a few signature flags G9...
* With an optional expiration date AS...
* With a checksum ORZGC3DBNRQQU over contextual data
  → Restrictions on remote user, domain
  → Restrictions on session identities
  → Restrictions on topic, subject

arpa2 guest identity

This is *a local alias for a remote user*

* Aliases under a userid like guest@example.com
* Signature-based: guest+GMZWG6ZLMN5SWY33FBI+@example.com
* Restricts remote ruser@rdomain but does not mention it
* Friendly to Realm Crossover, with consistent translations
* GMZWG6ZLMN5SWY33FBI may map back to ruser@rdomain
* Better aliases for email, XMPP, SIP, . . .

arpa2 selectors

ARPA2 Selectors are patterns for ARPA2 Identities.

ARPA2 Selectors are sets of ARPA2 Identities.

* john@example.com	matches just one identity
* @example.com	matches any user under a domain
* @.example.com	matches any user under any subdomain
* @.	matches any user under any domain

ARPA2 Selectors help us in Access Control

http://common.arpa2.net/md_doc_IDENTITY.html

arpa2 access control

˜@. %R ˜john@example.com %RW

* Access Control to a Document

* Access Control to a Local Identity (for Communication)

* Attributes can be set to modify semantics

* Rules are stored in a database or an application context

http://common.arpa2.net/md_doc_ACCESS_DOCUMENT.html

http://common.arpa2.net/md_doc_ACCESS_COMM.html

access control and signed identity

- We always check signatures on input
  - → . . . but will accept if none present
  - → . . . and require presence during Access Control
- We always add signatures on output
  - → . . . for which we may add a recipe while sending
  - → . . . and we skip signing without such a recipe

arpa2 groups

Like UNIX groupid. . . generalised to remote access

* Identities are as for users cooks@example.com
* Members add an alias cooks+johann@example.com
* Guests add a signature cooks+GEA2DGIBRGAQ+@example.com
* Incoming data: Map identities to group member
* Outgoing data: Map group member to delivery address
* Member selection: cooks+johann@example.com
* Member filter : cooks+johann+marie@example.com

This API is currently under development

arpa2 groups are local

Groups can welcome remote and local users alike

Members are addressed under the group and it’s domain
* SPF and DKIM never break on this scheme
* Services can use this for privacy by default
* Admins can break privacy to handle abuse

Better option than email forwarding:
* Define a group with the delivery address as a member
* Before email forwarding, make the sender a local guest

protocol: tls□kdh

* Kerberos authentication, ECDHE encryption
* Incorporate ticket key for Quantum Relief
* Extremely fast (like OpenSSH with Kerberos)
* See: draft-vanrein-tls-kdh

* Cryptographers: Rick van Rein, Tom Vrancken
* TA4NGI of daasi.de: Python CMS += TLS-KDH

protocol: http□sasl

Most protocol support SASL authentication, but not HTTP
* We add WWW-Authenticate: SASL
* Attributes realm, mech, c2s, s2c
* Integration with HTTP Status and such
* See: draft-vanrein-httpauth-sasl

protocol: kip

Encryption with Keyful Identity Protocol:
* Encrypt with a random sesion key
* ACL + session key $\rightarrow$ KIP $\rightarrow$ key mud
* Encrypted data + key mud $\rightarrow$ recipients
* SASL authentication + key mud $\rightarrow$ KIP $\rightarrow$ session key
* Decrypt data with session key
* https://gitlab.com/arpa2/kip/

Make keys available after authentication

Cut out the middle certificate

protocol: haan

To generate a free identity:
* HAAN Service holds a static key
* HAAN Service generates a random userid
* HAAN Service melts key and userid to a password
* HAAN Service sends domain, userid, password

To lookup a HAAN password for a SASL mechanism:
* HAAN Service holds a static key
* HAAN Service melts key and userid to a password
* HAAN Service lets any SASL mechanisms test the password

Have as many as you like – they're free

Bootstrap your online identity – without email

application: apache

A useful test platform; Henri and Rick make modules for:
* HTTP-SASL with a Cyrus-SASL2 backend
* HTTP-SASL with a Diameter backend (for Realm Crossover)
* HTTP userdir with a User: john header
* ARPA2 Access Control
* https://gitlab.com/arpa2/apachemod

Imagine an HTTP service with no accounts!

application: reservoir

Reservoir is our object store in standard protocols

* Metadata is searchable in LDAP
* ARPA2 Access Control (on Collections)
* Objectdata retrievable via any protocol
  → HTTP for Download links
  → AMQP for Inboxes and Outqueues
  → MSRP for Document Exchange during SIP calls
  → SFTP for Direct Access
* https://gitlab.com/arpa2/reservoir

Imagine a Reservoir store for your Project Group

application: haan

HAAN can be a Public Service run under a subdomain:
* Protocol stack: Diameter, SASL, HAAN
* Every public service has it own fixed key
* No need to store anything, low expected use
* Users can have many, stored with friends or lawyers
* Helps to build user freedom to move

HAAN is intended as fallback authentication:
* Realm Crossover for SASL is all you need
* The public service reports valid identity
* The rescued service can map the identity to a local one

application: yours!

We are hoping to get Realm Crossover into services
* We intend to modestly fund open source modules
* We prefer those with the highest impact
* Talk to us if you have an idea; hint others

Some projects may be more complex
* Kamailio with Realm Crossover? Might get hairy.
* Python http.* with Realm Crossover? Cool.
* SASL with Realm Crossover in your password manager? Nice.
* Maybe consider your own NGI Pointer project:
* https://pointer.ngi.eu/open-calls/

thanks for listening

* We want to given users control over their online presence

* We design this stuff for domain hosting parties

* We focus on core libraries, protocols, daemons

* We cannot do everything, even if we want to

  → We would love to see this in more applications

  → We can use support when proposing Internet Drafts info@openfortress.nl

http://openfortress.nl

OpenFortress*
digital signatures