# Stackable GSS Pseudo-Mechs

draft-williams-gssapi-stackable-pseudo-mechs-00

Nicolas.Williams@sun.com

60th IETF KITTEN BoF

# History

- 2000: LIPKEY [RFC2847], basic-over-SPKM
- Early-2003: CCM-BIND (I-D), first stackable GSS-API pseudo-mechanism
- 58th IETF: hallway discussion of mechanism stacking resulted in:
  - Need for abstraction
  - Ideas for other stackable pseudo-mechs
  - Need to think about negotiation, complexity
- 60th IETF: 1st I-D on stackable pseudo-mechs

# Glossary

- Concrete mechanism
  - A GSS-API mechanism that can be used as is
- Pseudo-mechanism
  - A GSS-API mechanism that cannot be used without reference to a concrete mechanism; e.g., SPNEGO
- Stackable pseudo-mech
  - A mechanism that is to be “stacked above” or combined with a composite or concrete mechanism
- Composite mechanism
  - A combination of a stackable and a composite or concrete mech

# Introduction

- The GSS-API is a generic interface to security mechanisms
  - Mechanisms are addressed by their OIDs
  - Mechs define: context tokens, per-msg context tokens, and sundry GSS details, such as name forms
- GSS mechanisms exist for: Kerberos V, PKIX (SPKM), and others, such as Microsoft's NTLMSSP, Sun's mech_dh
- GSS pseudo-mechanisms exist for: negotiating mechanisms (SPNEGO)

# Introduction (cont.)

- In the process of developing a new lightweight GSS-API pseudo-mechanism for NFS we expanded on the GSS-API notion of channel bindings and the new mechanism (CCM-BIND) came to be about channel bindings
- At the same time we developed the notion of GSS mechanism stacking so we could leverage existing GSS mechanisms in the construction of new ones
  - CCM-BIND being one example

# Introduction (cont.)

- Composite mechanisms have OIDs, just like any other mechanism
  - Composite mech OIDs are made by prefixing the OID of the stackable mechanism to that of the mechanism stacked below it
- Stackable mechs can be stacked over other composite mechs, making a stack
- Composite mechs are used just like concrete mechs

# LIPKEY: Almost a Stack

- LIPKEY is a GSS mechanism that does the SPKM equivalent of basic-over-SSL
  - LIPKEY first uses SPKM-3 to establish a security context that authenticates the acceptor (using its cert) but not the initiator
  - then it sends the initiator's name and password confidentiality protected with the SPKM-3 context
- But LIPKEY is **not** an example of a stackable pseudo-mech, though it could have been
  - No OID prefixing; LIPKEY only works over SPKM

# Ideas for Stackable Pseudo-Mechs

- Proper channel binding and negotiation
  - CCM-BIND
- PFS
- Compression
- Basic-over-*
- Three-party authentication
- etc...

# Example: PFS

- Let's call this the PFSMECH
- PFSMECH context tokens might contain:
  - Context tokens for mech stacked below
  - DH public parameters
- PFSMECH would have its own per-msg tokens
  - Perhaps based on existing design, such as krb5's
- One PFSMECH OID prefix per-{group, ciphers}? Or other scheme?
  - This would eschew GSS-API lameness w.r.t. QoPs

# Problems

- Not all mechanism stacks will make sense
  - {pfs, compress, krb5} is no good, but {compress, pfs, krb5} is Ok
- Complexity
  - Many valid composites
  - How to negotiate mechanisms?
- GSS_Indicate_mechs() and friends

# Problems (cont.)

- Security analysis of composite mechanisms
  - What combinations make sense, which don't?
  - What are the attributes of a composite mechanism?

# Solutions

- GSS_Indicate_mechs() and friends MUST NOT indicate stackable mechs
- GSS_Indicate_mechs() and friends MUST NOT indicate composite mechs unless explicitly configured to do so (and even then...)
- Add new APIs for indicating stackable/ composite mechs

# Solutions (cont.)

- Users of composite mechs know what features they want from them, but why should they know the OIDs of the composite mechs they need?
  - Add APIs for inquiring mechs for/by their attributes
- These new APIs are all OPTIONAL
  - Without them apps have to hardcode composite mech OIDs – no big deal
- Mechanism attributes have OIDs and symbolic names (GSS_C_MA_*)

# Solutions (cont.)

- Stackable pseudo-mechanism specifications should describe
  - Constraints on mechanisms, by attributes, that can be stacked below
  - How to compute the attributes of mechanisms composed with them in terms of the attributes of the mechanisms stacked below

# Benefits of the New APIs

- No need to hardcode mechanism OIDs anymore
  - e.g., SSHv2 implementations MUST NOT use SPNEGO, but SPNEGO might get new OIDs[*]
    - Let SSHv2 implementations query for/by mechanism attributes and ignore any mechs that negotiate mechs
- Mechanism attributes give us a way to formalize the descriptions of mechanisms
  - Hardcoding attrs' symbolic names is better than hardcoding mechanism OIDs; see above

# Benefits of the New APIs (cont.)

- Indicating mechs by attributes makes GSS-API applications more general
  - Unless the new mech-specific GSS-API extensions

# New APIs

- GSS_Indicate_mechs_by_attrs()
- GSS_Inquire_mechs_for_attrs()
- GSS_Display_mech_attr()
- [utility] GSS_Compose_OID()
- [utility] GSS_Decompose_OID()
- GSS_Indicate_negotiable_mechs()
- GSS_Negotiate_mechs()

# Mechanism Attributes

- Concrete, stackable, composite, glue[*], other
- Deprecated (e.g., old krb5 mech OID), non-standard (e.g., GSI's SSL mech)
- Authenticates initiator, acceptor, both, neither
- Supports credential delegation
- Supports confidentiality and/or integrity protection, replay, out-of-sequence detection
- PFS, channel bindings, compression
- Etc...

# Mechanism Attributes (cont.)

- GSS_C_MA_*
- Mech attrs are identified by symbolic names **and** OIDs
  - So that SET OF mechanism attributes is SET OF OBJECT IDENTIFIER
  - Which leverages existing C-Bindings for OID sets
    - Keeps the API simple, stupid

# Internet-Drafts

- draft-ietf-nfsv4-ccm-02.txt
- draft-williams-gssapi-stackable-pseudo-mechs-00.txt

# Q/A

- Questions?
- Please review