IETF 56
San Francisco

NFSv4-wg

March 18, 2003

XDR, SECINFO, CCM

Mike Eisler

Network Appliance

XDR

- draft-ietf-nfsv4-xdr-bis-00.txt
- XDR is currently a Draft Standard
- Purpose of i-d is enable advancement to Standard by adding required sections (IANA, copyright, etc.)
- For changes look at http://www.eisler.com/nfsv4-wg/draft-ietf-nfsv4-xdr-bis-00.cb
- Intend to seek wg last call next week

SECINFO Problems – parent dir

- draft-ietf-eisler-nfsv4-secinfo-00.txt
- In NFSv4.0, SECINFO takes a directory filehandle plus a component name
  - An analog of LOOKUP
- Limits ability to determine security of parent directory (since parent dir lookup uses LOOKUPP, not LOOKUP "..")
- Effectively means that in NFSv4.0, LOOKUPP must always be honored even if parent has different security flavor than child!

SECINFO Problems – how to cross a security domain

- Example: /foo is exported sec=sys
  - /foo/bar is exported sec=krb5
  - Client has filehandle for /foo (fhFoo)
  - Client does **{ PUTFH fhFOO, LOOKUP "bar" },** using sec=sys
  - Returned NFSv4ERR_WRONGSEC
  - Client does SECINFO fhFoo "bar" and is told to use krb5.
  - Client issues **{ PUTFH fhFOO, LOOKUP "bar" },** using sec=krb5
    - Should server reject PUTFH with NFS4ERR_WRONGSEC since security mismatches that required for /foo?

SECINFO Problems – PUTFH returns NFS4ERR_WRONGSEC

- NFSv4.0 specifications allows PUTFH operation to return NFS4ERR_WRONGSEC
- Specification says client should issue a SECINFO using the parent filehandle and component name of the filehandle PUTFH was used with.

SECINFO Problems – Support for other flavors

- The specification doesn't mention how to encode SECINFO results for flavors other than AUTH_NONE, AUTH_SYS, RPCSEC_GSS

SECINFO Problems – PUTFH (continued)

/foo/bar exported as sec=krb5
client -> < -server

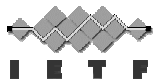
fhBar obtained on a previous OPEN

```
PUTFH fhBar READ      ->
                  <     - WRONGSEC
```

What if client doesn't record filehandle of "/foo", and fhBar's component name ("bar")?

Proposed fixes to SECINFO problems – other flavors

- I-d makes it clear that flavors other than AUTH_NONE, AUTH_SYS, and RPCSEC_GSS are supported
- Technically, SECINFO results are specific to a given flavor
- In practice, RPCSEC_GSS is the only one of importance that has flavor specific content
- NFSv4.0 implementations should assume this is the case.
- AUTH_DH (AUTH_DES) lives!



Proposed fixes to SECINFO problems – PUTFH and LOOKUPP

- New operation SECINFO_NO_NAME
- As implied from operation's name, no component name passed
- SECINFO_NO_NAME has two styles:
  - *Style = current* queries security info of current filehandle
  - Style = parent queries security info of parent directory

Proposed fixes to SECINFO problems – crossing security domain

- I-d clarifies that if the server wants to allow crossing of security domains (**{ PUTFH fhFOO, LOOKUP “bar” }),** then it is permissible to allow the PUTFH and return NFS4ERR_WRONGSEC for the LOOKUP.

SECINFO – next steps

- Make SECINFO i-d an NFSv4-wg work item
- Update i-d to note that domain crossing issue applies to
  - PUTFH+OPEN as well as PUTFH+LOOKUP
  - { PUTROOTFH, PUTPUBFH } X { LOOKUP, OPEN }
- Recast SECINFO changes into an NFSv4.1i-d
  - Need document editor for NFSv4.1!

CCM – Credential Cache Mechanism

- draft-eisler-nfsv4-ccm-00.txt
- Problem Statement: As network media get faster, the overhead of encryption, integrity, and even plain authentication in RPCSEC_GSS will impede performance
- CCM is a GSS mechanism that allows a client user and server to authenticate each other once

CCM – Theory of Operation

- If client and server believe that channel is secure (e.g. protected with IPsec, SSH, ...), then NFSv4 server and client negotiate “down” to use CCM, via NFS4ERR_WRONGSEC, and SECINFO.
- CCM returns zero bytes for gss_get_mic(), and returns the same input for gss_wrap().

CCM – Changes since version 00 of i-d

- Nicholas Williams proposed that CCM be wrapper mechanism around real mechanisms, such as Kerberos V5 (RFC1964)
  - This makes it easier to integrate (implement) into existing GSS-API frameworks

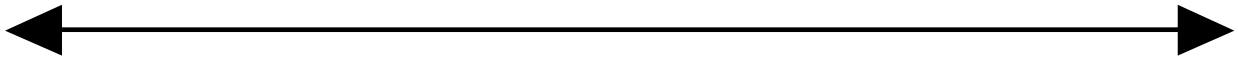
CCM – Issues

- Several people believe that CCM MUST specify channel bindings which bind a CCM context to a specific session key of the underlying channel
  - This provides end to end security at the RPCSEC_GSS level and at the secure channel level
- However, at least one upper layer protocol, iSCSI does not require end-to-end secure channel. This is a matter of user policy

CCM – What an attacker can do without channel bindings

3. Application client and server authenticate each other, but are unaware attacker tampers with traffic.

2. Attacker in the middle tampers with key exchange. Each end has a different session key



1. Secure channel protected with unauthenticated DH key exchange

CCM – Attacker defeated by channel bindings

4. App client and server authenticate each other, including a checksum of session key. Mismatch is detected

3. Application client and server ask secure channel for the session key.

2. Attacker in the middle tampers with key exchange. Each end has a different session key

1. Secure channel protected with unauthenticated DH key exchange

CCM – Issues (continued)

- Channel bindings are very nice, but
  - What if there are no APIs to allow application to inquire of secure channel layer (e.g. IPsec, SSH) of session key information?
    - Session-key based channel bindings are not defined for IPsec, SSH, etc., so initially no API support is likely
  - What if there is a firewall/NAT box between client and server, such that there are two IPsec secure channels?
    - Channel bindings get in the way

CCM – My Current Thinking on Channel Bindings

- Channel bindings ought to be a SHOULD, and not a MUST, since not all implementors (whether NFS, or something else) control the secure channel implementation
- CCM implementations SHOULD let users set policy on channel bindings
- CCM probably needs a way to negotiate the use of channel bindings

CCM – Issues (continued)

- The scenario of an unauthenticated secure channel seems to be somewhat contrived.
  - On the other hand, if something like CCM w/ channel bindings for applications is ubiquitous, then unauthenticated IPsec is very convenient
- iSCSI deals with the channel bindings issue by declaring it to be a policy decision on the part of the user.