NetApp™

Go further, faster™

Kerberized NFS

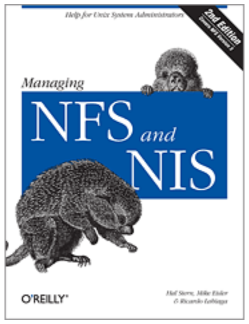
2010 Kerberos Conference

Ricardo Labiaga

Mike Eisler

About Us

- Co-Authors (with Hal Stern) of Managing NFS and NIS, 2nd Edition



- NFS developers since 1990s and 1980s
- Currently leading NFS development at NetApp for Linux and NetApp's storage development operating system, Data ONTAP

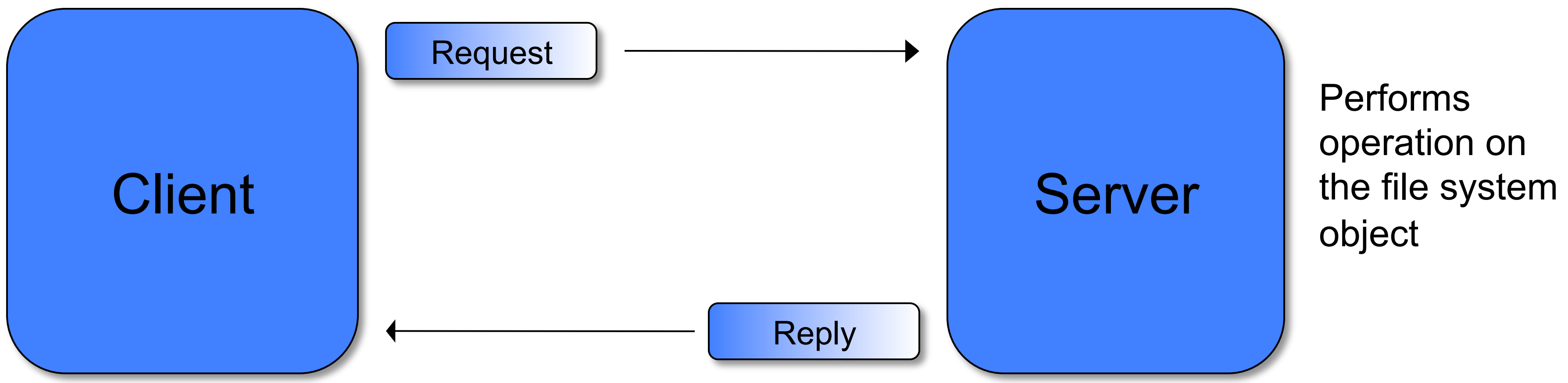
NetApp and Kerberos

NetApp and Kerberos

- Network Attached Storage
  – NFS Server Authentication
  – CIFS Server Authentication
- NetApp Applications
  – Snap Manager for Oracle
  – Snap Manager for Exchange
  – Snap Manager for Virtual Infrastructure
  – Many more

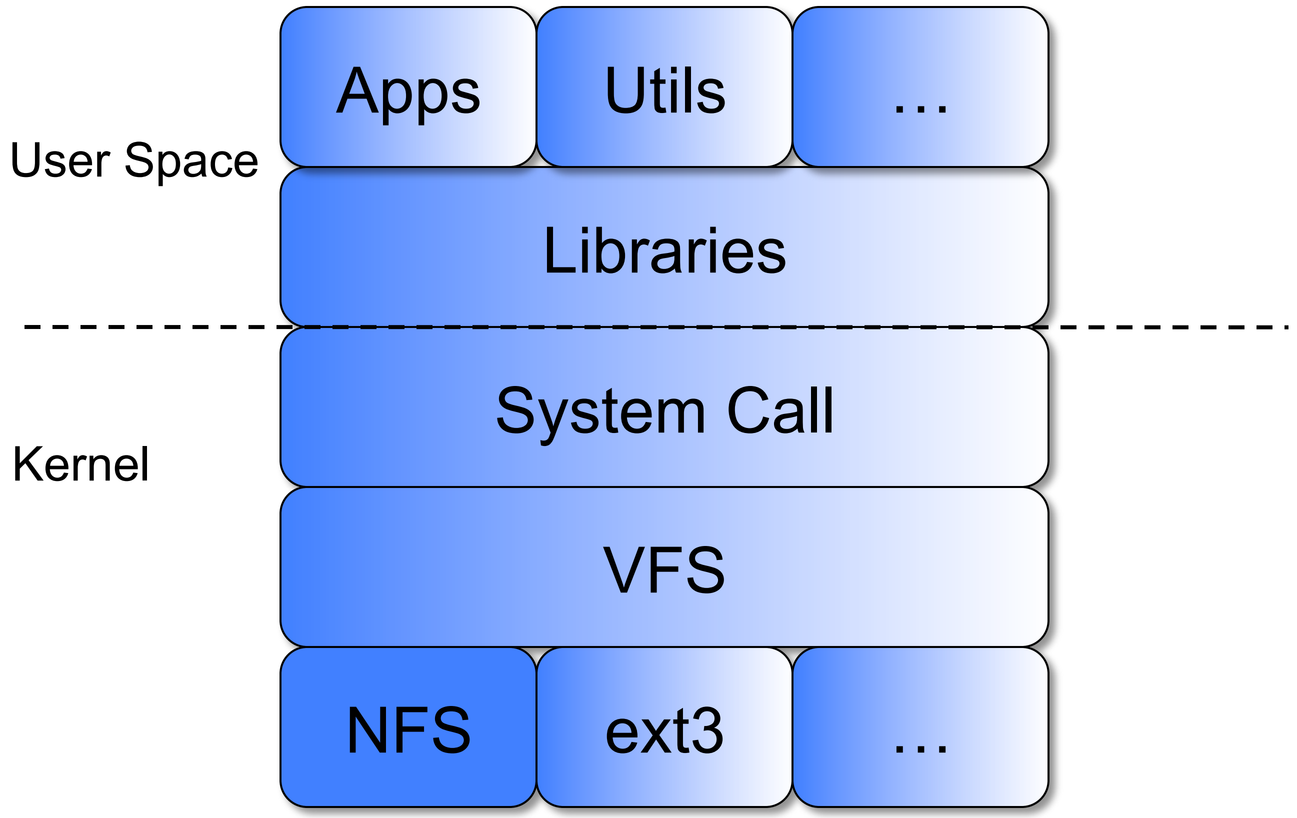
NFS in a Nutshell

Client / Server Message Passing

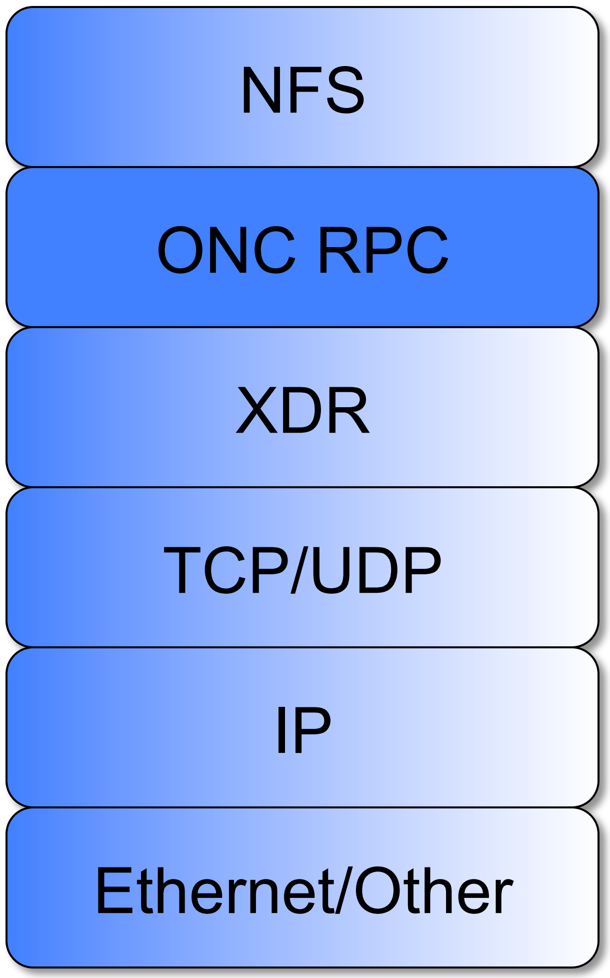


Performs operation on the file system object

NFS Operating System Layering



NFS / RPC Protocol Layering



NFS Authentication

NFS Authentication without Kerberos

- What authentication?
- AUTH_SYS RPC security flavor
  - No crypto
    - predates the relaxation of U.S. export controls that permitted crypto for authentication
- This is trivial

```
client% su
client# su – alice
alice% cd ~/
alice% cd private
```

  - Kerberized NFS prevents this, because without Kerberos credentials super-user cannot access data protected with strong permissions

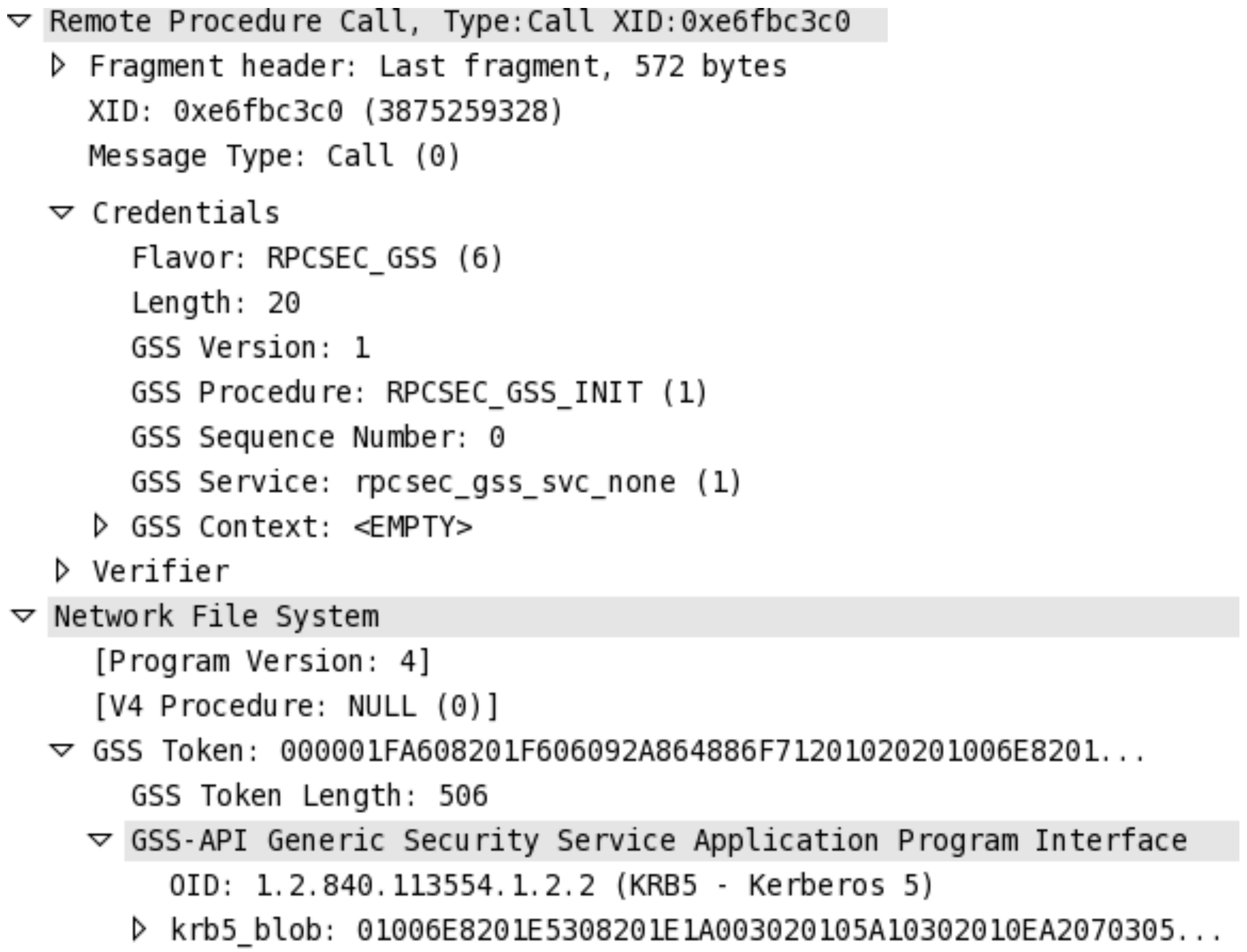
How NFS Integrates with Kerberos

- RPC is responsible for authentication
- RPC supports multiple security flavors
- Flavor number 6 is RPCSEC_GSS (RFC 2203)
  - A “network transport” for the Generic Security Services API (GSS-API).
- Kerberos as a GSS-API mechanism “plugs” into RPCSEC_GSS
- RPC requests carry a credential and verifier
  - Each up to 400 bytes
- RPC replies carry a verifier
  - Up to 400 bytes

RPCSEC_GSS
Context Initialization Phase

- RPC client invokes GSS_Init_Sec_Context
- Output token sent to RPC server
- GSS context can be too large for RPC credential
  – Consequently, RPC arguments are overloaded to carry context tokens

GSS Context Init Request

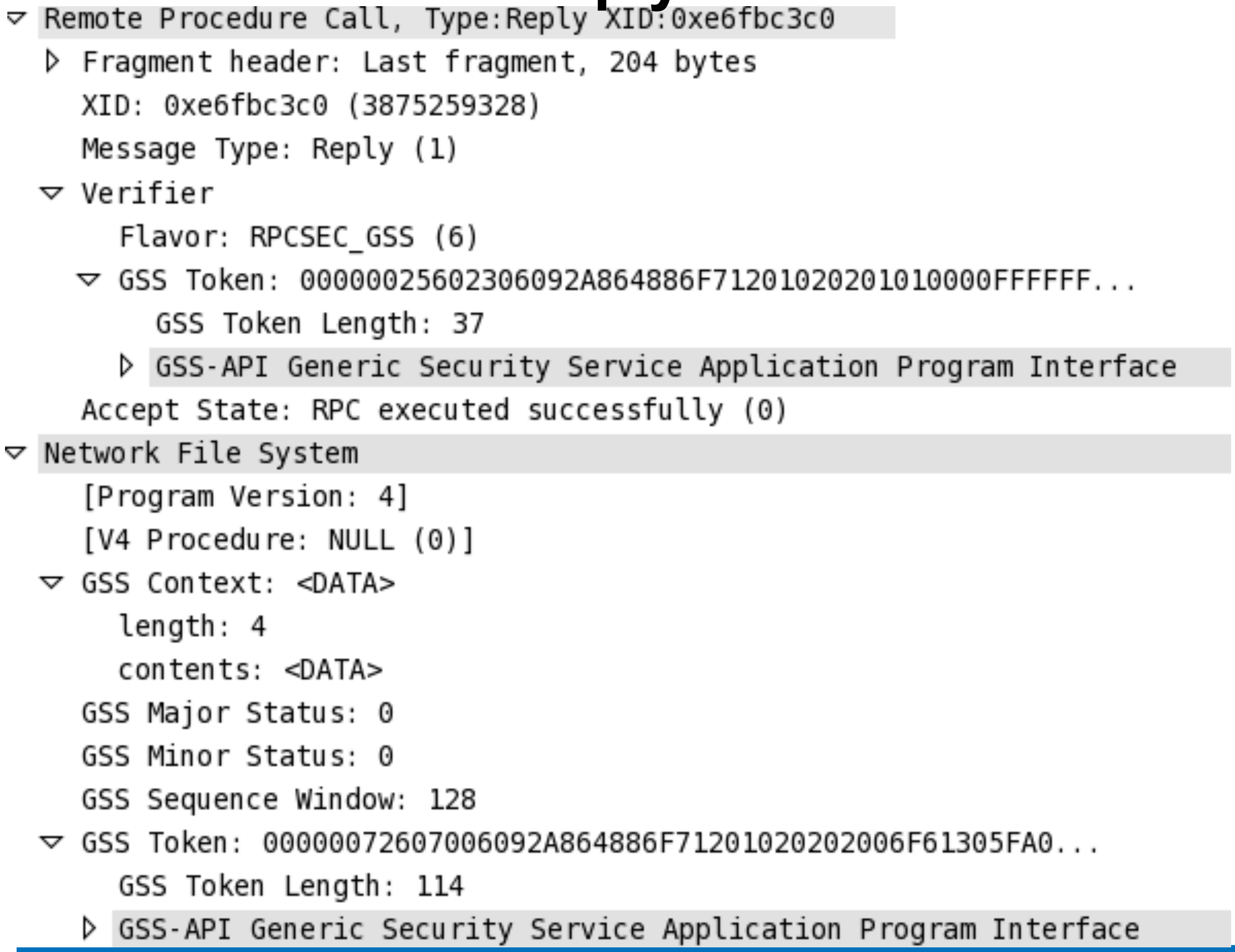


```
▽ Remote Procedure Call, Type:Call XID:0xe6fbc3c0
  ▷ Fragment header: Last fragment, 572 bytes
    XID: 0xe6fbc3c0 (3875259328)
    Message Type: Call (0)
  ▽ Credentials
      Flavor: RPCSEC_GSS (6)
      Length: 20
      GSS Version: 1
      GSS Procedure: RPCSEC_GSS_INIT (1)
      GSS Sequence Number: 0
      GSS Service: rpcsec_gss_svc_none (1)
    ▷ GSS Context: <EMPTY>
  ▷ Verifier
▽ Network File System
    [Program Version: 4]
    [V4 Procedure: NULL (0)]
  ▽ GSS Token: 000001FA608201F606092A864886F71201020201006E8201...
      GSS Token Length: 506
    ▽ GSS-API Generic Security Service Application Program Interface
        OID: 1.2.840.113554.1.2.2 (KRB5 - Kerberos 5)
      ▷ krb5_blob: 01006E8201E5308201E1A003020105A10302010EA2070305...
```

RPCSEC_GSS
Context Initialization Phase (cont…)

- RPC server invokes GSS_Accept_Sec_Context
- Output token sent to RPC client in RPC reply
- GSS context can be too large for RPC verifier
  - Consequently, RPC results are overloaded to carry context tokens

GSS Context Init Reply



```
▽ Remote Procedure Call, Type:Reply XID:0xe6fbc3c0
  ▷ Fragment header: Last fragment, 204 bytes
    XID: 0xe6fbc3c0 (3875259328)
    Message Type: Reply (1)
  ▽ Verifier
      Flavor: RPCSEC_GSS (6)
    ▽ GSS Token: 0000002560230609 2A864886F71201020201010000FFFFFF...
        GSS Token Length: 37
      ▷ GSS-API Generic Security Service Application Program Interface
    Accept State: RPC executed successfully (0)
▽ Network File System
    [Program Version: 4]
    [V4 Procedure: NULL (0)]
  ▽ GSS Context: <DATA>
      length: 4
      contents: <DATA>
    GSS Major Status: 0
    GSS Minor Status: 0
    GSS Sequence Window: 128
  ▽ GSS Token: 00000072607006092A864886F71201020202006F61305FA0...
      GSS Token Length: 114
    ▷ GSS-API Generic Security Service Application Program Interface
```

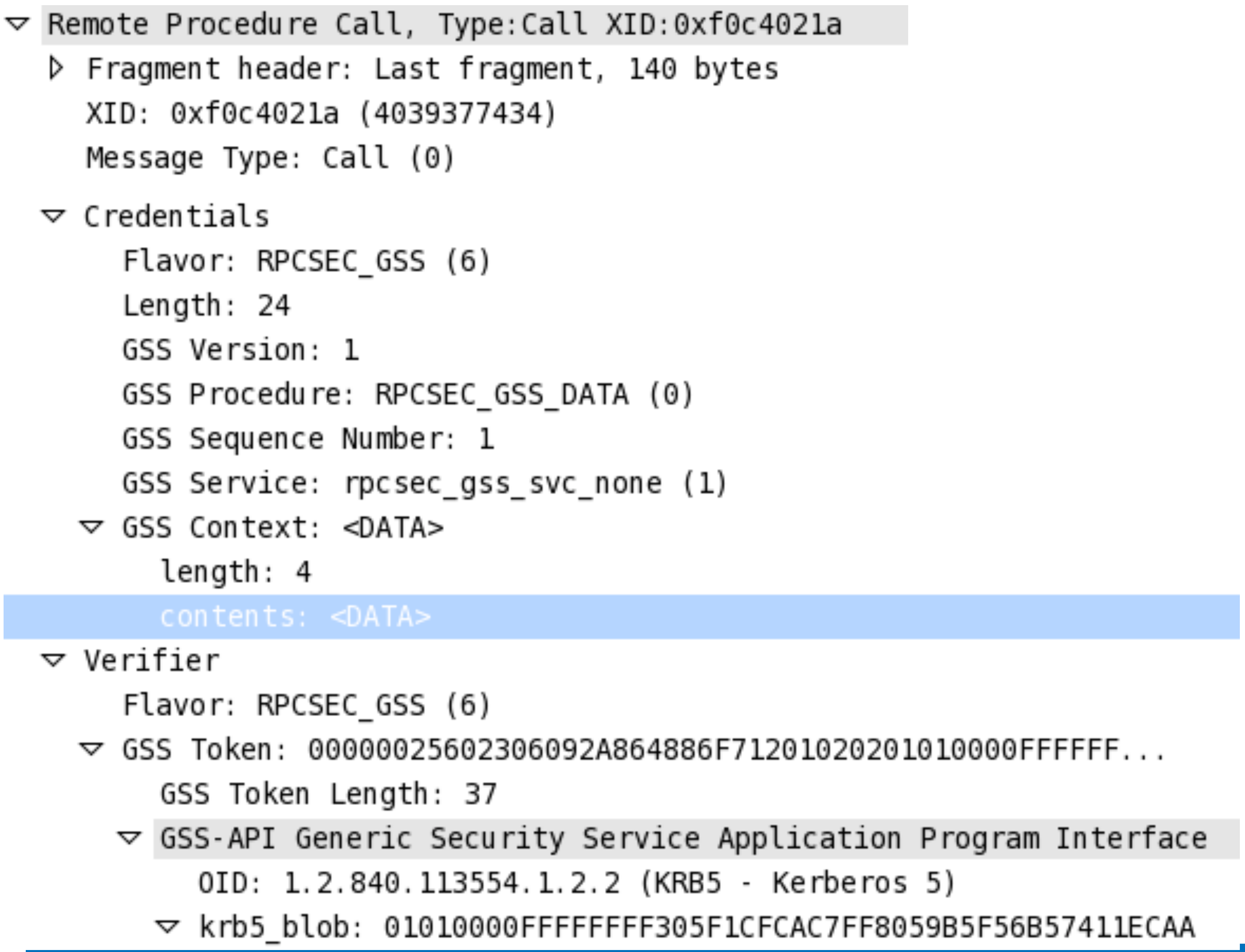
RPCSEC_GSS
Context Initialization Phase (cont…)

- RPC exchanges continue until GSS status is GSS_COMPLETE

RPCSEC_GSS
Context Established Phase

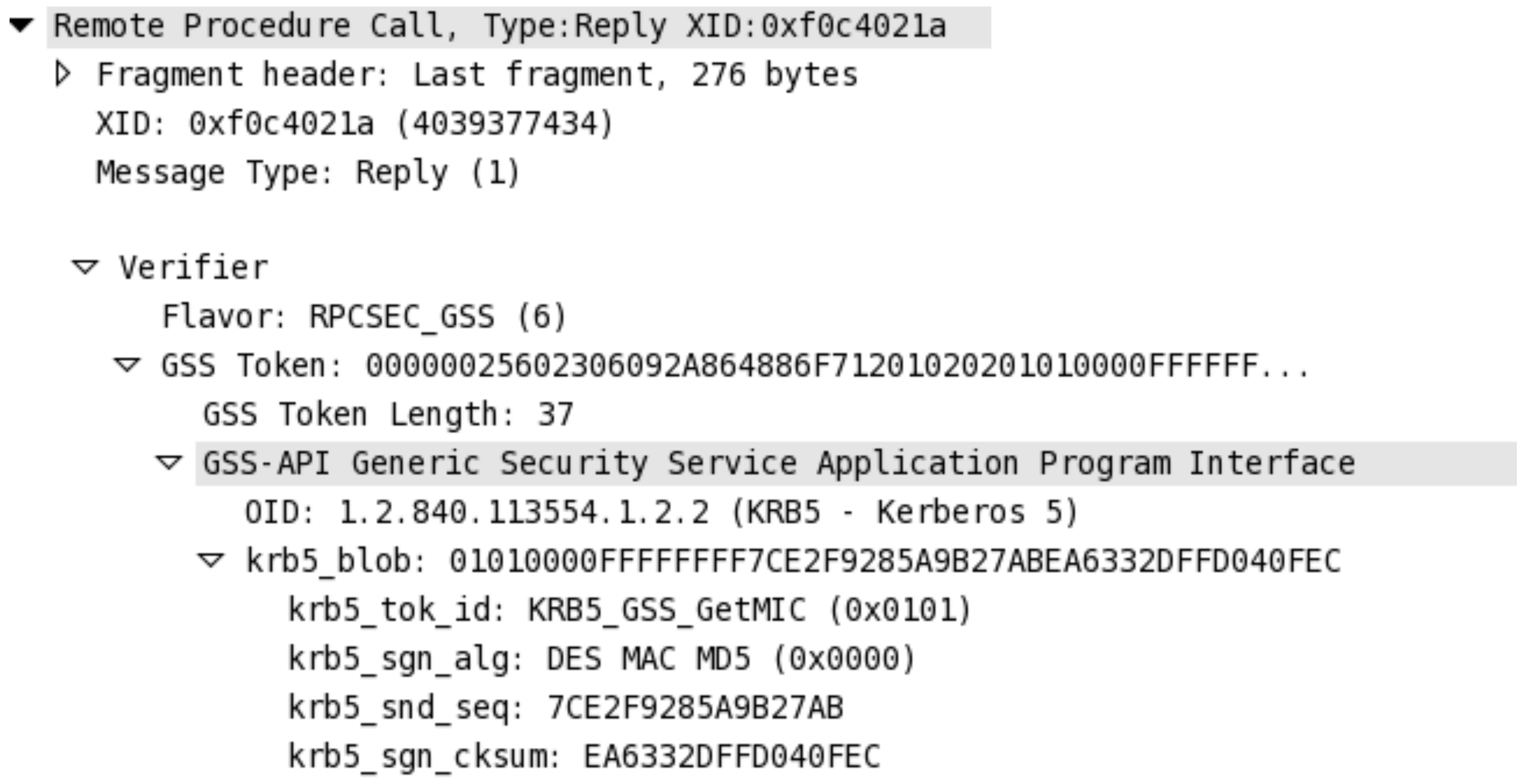
- Request
  - Credential
    - identifies established GSS context
    - contains a sequence number
  - Verifier contains MIC of RPC header (including credential)
- Reply
  - Verifier contains MIC of sequence number

NFS Request



```
▽ Remote Procedure Call, Type:Call XID:0xf0c4021a
  ▷ Fragment header: Last fragment, 140 bytes
    XID: 0xf0c4021a (4039377434)
    Message Type: Call (0)
  ▽ Credentials
      Flavor: RPCSEC_GSS (6)
      Length: 24
      GSS Version: 1
      GSS Procedure: RPCSEC_GSS_DATA (0)
      GSS Sequence Number: 1
      GSS Service: rpcsec_gss_svc_none (1)
    ▽ GSS Context: <DATA>
        length: 4
        contents: <DATA>
  ▽ Verifier
      Flavor: RPCSEC_GSS (6)
    ▽ GSS Token: 00000025602306092A864886F71201020201010000FFFFFF...
        GSS Token Length: 37
      ▽ GSS-API Generic Security Service Application Program Interface
          OID: 1.2.840.113554.1.2.2 (KRB5 - Kerberos 5)
        ▽ krb5_blob: 01010000FFFFFFFF305F1CFCAC7FF8059B5F56B57411ECAA
```

NFS Reply



```
▼ Remote Procedure Call, Type:Reply XID:0xf0c4021a
  ▷ Fragment header: Last fragment, 276 bytes
    XID: 0xf0c4021a (4039377434)
    Message Type: Reply (1)

  ▽ Verifier
      Flavor: RPCSEC_GSS (6)
    ▽ GSS Token: 00000025602306092A864886F71201020201010000FFFFFF...
        GSS Token Length: 37
      ▽ GSS-API Generic Security Service Application Program Interface
          OID: 1.2.840.113554.1.2.2 (KRB5 - Kerberos 5)
        ▽ krb5_blob: 01010000FFFFFFFF7CE2F9285A9B27ABEA6332DFFD040FEC
            krb5_tok_id: KRB5_GSS_GetMIC (0x0101)
            krb5_sgn_alg: DES MAC MD5 (0x0000)
            krb5_snd_seq: 7CE2F9285A9B27AB
            krb5_sgn_cksum: EA6332DFFD040FEC
```

RPCSEC_GSS Context Established Phase

- Why not use GSS_GetMIC's sequencing?
  - RPC can work over connectionless transports, so lost requests/replies are not infrequent
  - GSS sequencing would force serialization
    - RPC needs to have multiple outstanding requests from the same { src port, src addr, dst port, dst addr } tuple

RPCSEC_GSS Context Established Phase

- Optional integrity
  - MIC of arguments + sequence number
  - MIC of results + sequence number

- Optional privacy
  - WRAP of arguments + sequence number
  - WRAP of results + sequence number

RPCSEC_GSS + Kerberos: Impact on NFS Industry

- NFSv4 (an IETF standard) mandates RPCSEC_GSS w/ Kerberos V5 for implementation that claim conformance to NFSv4 standard
- Mandate not extended to use
- Every NFSv4 client and server supports Kerberos V5 authentication
- Linux, BSD, and every major UNIX has Kerberized NFS

Support of this technology is broad (except for new entrants to storage business)

NetApp™

However …

- Actual customer use is weak
- As measured by NetApp AutoSupport:
  - fewer than 1% of systems use Kerberized NFS
  - just one customer (albeit 100s of systems) deploys Kerberized NFS for majority of NFS use.

Why is Kerberized NFS Unpopular?

- We can learn lessons from Windows 2000
  - "Domain Controller" made Kerberos setup a hidden and all but mandatory part of domain operations
    - domain creation creates a KDC
    - adding a user or machine creates a principal in KDC
- Outside the Windows world, Kerberos setup is manual and complex task
  - Manual tasks often go wrong
  - MIT Kerberos does not provide much insight when something goes wrong

Typical Kerberized NFS Problem Report

The implementation of NFS4 without GSS for the server and the client was really easy and is still up and running well.
But the implementation of the GSS API is faulty in our environment. We read a lot of staff in the internet and tried several tutorials. But the result is always the same: "mount.nfs4: Permission denied".
On details:
After the mount command (with -vvv) we got the message: mount: pinging: prog 100003 vers 4 prot tcp port 2049
And after 25 sec we got the message: mount.nfs4: Permission denied
Also we found the drawing: http://www.citi.umich.edu/projects/nfsv4/gssd/, which was really helpful.
So, with full logging on rcp and nfs (echo 32767 > /proc/sys/sunrpc/rpc_debug, echo 65535 > /proc/sys/sunrpc/nfs_debug, imapd verbose level on 10, gssd and scvgssd with -vvv) and tcpdump,we followed the drawing.
We could see the line 3 and 6 in the TCPDUMP (but are a little bit wondering, while this happens twice):

As output of the gssd we got on the client:

```
Sep  7 13:10:11 xxxkernel: <-- nfs4_set_client() = 0 [new ffff81012e280800]
Sep  7 13:10:11 xxxkernel: --> nfs4_init_server()
Sep  7 13:10:11 xxxkernel: <-- nfs4_init_server() = 0
Sep  7 13:10:11 xxxkernel: --> nfs4_path_walk(,,/)
Sep  7 13:10:11 xxxkernel: RPC:    0 new task procpid 20373
Sep  7 13:10:11 xxxkernel: RPC:    0 allocated task
Sep  7 13:10:11 xxxkernel: RPC:    0 looking up RPCSEC_GSS cred
Sep  7 13:10:11 xxxkernel: RPC:      gss_create_cred for uid 0, flavor 390003
Sep  7 13:10:11 xxxkernel: RPC: gss_upcall for uid 0
Sep  7 13:10:11 xxxkernel: RPC:      gss_find_upcall found nothing                    <<<<<<<<<<---------- That really looks faulty
Sep  7 13:10:11 xxxkernel: RPC: 1307 freeing task
Sep  7 13:10:36 xxxrpc.gssd[8033]: WARNING: Failed to create krb5 context for user with uid 0 for server xxx.yyy.de
Sep  7 13:10:36 xxxrpc.gssd[8033]: WARNING: Failed to create krb5 context for user with uid 0 with credentials cache MEMORY:/tmp/
krb5cc_machine_ttt.yyy.de for server xxx.yyy.de
Sep  7 13:10:36 xxxrpc.gssd[8033]: WARNING: Failed to create krb5 context for user with uid 0 with any credentials cache for server xxx.yyy.de
Sep  7 13:10:36 xxxrpc.gssd[8033]: doing error downcall
```

On the server with the scvgssd, we got nothing in the log (/var/log/messages), even so the logging is also set to maximum.
Also we did a trace with netmon, which is attached. There is a 5th transmission, which we haven`t seen with tcpdump-
Do you have an idea, what we could do to make this running? Google shows us the sourcecode of the gssapi, when we search this error ;-)

Note on Kerberos and Solaris

- Solaris also de-couples Kerberos set up
- However, Solaris customer visible documentation provides very precise recipes for performing each action:
  - joining NFS client to KDC realm
  - joining NFS server to KDC realm
  - add user to KDC realm
- Recipes can be cut from documentation and pasted into shell (client, server, KDC)
  - Those who cut and paste succeed nearly 100% of the time
  - Those who do not fail nearly 100% of the time

How Kerberos in non-Windows World Could Work

- Link LDAP/NIS set up with Kerberos
- Extend LDAP and NIS setup to automatically set up Kerberos
  - Adding hosts to directory should add Kerberos machine credentials to KDC
  - Machine credentials should be automatically populated on hosts
  - Adding users to directory should add users to KDC
- Design a new protocol to re-populate machine credentials when they change

FreeIPA Release 2 looks promising

- Free Identity, Policy and Audit integrated management solution
- Provides Directory Server (389), MIT Kerberos, NTP, DNS (BIND), on Linux
  - Installation configures all components: NTP, Directory Server, KDC, Apache, GUI, and client side bits
- http://www.freeipa.org/page/Main_Page

The MIT Consortium can help

- Of course every Linux distributor, UNIX vendor, and NFS storage vendor owns this problem
- However unique solutions for each Linux and UNIX distribution will make interoperability more challenging
- Storage vendors cannot dictate directory services and authentication integration
  - However storage vendors will follow such integration
  - Even non-Windows CIFS servers have painless Kerberos (Active Directory) set up

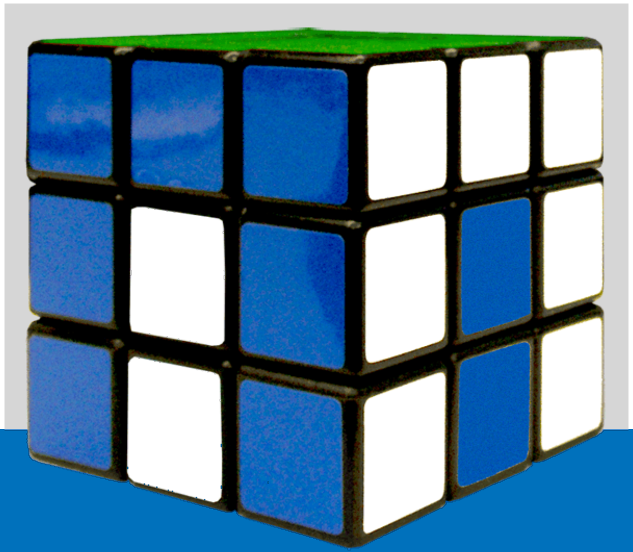
Opportunity for Microsoft

- Make Active Directory a much more attractive solution to customers that deploy lots of non-Windows-based hosts
  - Even shops of UNIX/NFS bigots have some Windows

What if nothing is done?

- Most NFS traffic will continue to be unsecured
  - Customers will continue to rely on physical network security
- If black hats ever get tired writing for malware for Windows
  - Eventually one will realize it is easy to produce malware that exploits AUTH_SYS as a vector
    - Easier to produce than the Morris Worm of 1988

Thanks



Questions?