Samba XP 2009

Samba and Likewise RPC testing and comparing the implementations

Rafal Szczesniak
Agenda

• Part I
  – Overview
  – Testing environment
  – Limitations
  – A few \samr tests

• Part II
  – Creating new rpc client and server
  – Differences between Samba and LWISO
  – Testing simple rpc interface
  – Conclusions
Part I: Well-known rpc interfaces
Brief overview

Samba rpc implementation

- started when WinNT domains appeared (over 10 years ago!)
- based on scraps of information from the internet and a lot of time spent with tcpdump
- hand-marshalled (initially)
- idl-based since Samba 4 showed up
- idl-based also in Samba 3 thanks to Guenther Deschner's great effort
Brief overview

Samba 4

- lsad, samr (…) client
- rpc_server
- librpc/idl and librpc/gen_ndr
- librpc/ndr and librpc/rpc
- libcli
- smb_server
Likewise Open (LWISO) implementation

- client side started in late 2007
  - based on original DCE/RPC framework (publically released by Novell)
  - originally employing libsmbclient library
  - new implementation of smb client started in September last year
  - complete named pipes support added to DCE/RPC in December
  - now employing LWIO
Brief overview

LWISO rpc implementation

- server side started with LWIO
  - original DCE/RPC framework doesn't support named pipes
  - LWIO includes named pipes server driver (npvFs)
  - this enables a server to listen on named pipe
Brief overview

LWISO

librpc/lsa, librpc/samr, (…)

librpc/idl

lsass/server/rpc/lsa,
lsass/server/rpc/samr, (…)

lsass/server/rpc/idl

dcerpc

lwmsg

lwio/server/rdr

lwio/server/npvfs

lwio/server/srv

© Copyright 2009 Likewise
Testing environment

1) Ubuntu 7 with:
   1) Samba 4 provisioned as DC
   2) DNS server – primary for Samba 4
2) Ubuntu 8 with:
   1) LWISO joined to AD
3) Windows Server 2003 configured as:
   1) DC and primary DNS for (2)
   2) secondary DNS for (1)
Limitations

Samba configuration
- DC using krb5 for authentication
- DNS server running on the same system
- secondary DNS on windows server so all requests can be resolved asking the same server
- LWISO implements very limited set of rpc calls so far, so limited test has to be used too

Testing tool: smbtorture
Limitations

LWISO configuration
- there is no security checks yet, so this is only a test of NDR correctness
- security descriptor library is already implemented
- the missing part is passing authentication info to server side of DCE/RPC runtime which enables creating access token

Testing tool: testrpc
Testing environment

Samba 4

ubuntu.ubuntu7.net

DC: UBUNTU7.NET

LWISO

ubuntu8-desktop.titanianet.net

member: TITANIANET.NET

testrpc

smbtorture

Win 2003 srv

titania.titanianet.net

DC: TITANIANET.NET

TGT for admin@TITANIANET.NET

TGT for administrator@UBUNTU7.NET
Something simple for start – connect and enumerate domains

SamrConnect[2-5]
SamrEnumDomains
SamrClose

smbtorture: LWIS-SAMR-DOMAINS test
testrpc: SAMR-DOMAINS test
Running tests

One step further – query each domain info levels
SamrConnect[2-5]
SamrEnumDomains
SamrOpenDomain
SamrQueryDomainInfo
SamrClose

smbtorture: LWIS-SAMR-DOMAINS-QUERY test
testrpc: SAMR-DOMAINS-QUERY test
Running tests

More complexity – enumerate local users and groups
SamrConnect[2–5]
SamrEnumDomains
SamrOpenDomain
SamrQueryDomainInfo
SamrEnumUsers, SamrEnumDomAliases
SamrClose
smbtorture: LWIS-SAMR-USERS-ENUM test
testrpc: SAMR-USERS test
Part II: Sample testing rpc service
Creating something new

How difficult is it to create a new rpc client and server in Samba?

- create idl file and place it in librpc/idl
- add new SUBSYSTEM in librpc/config.mk
- optionally add torture test and modify torture/config.mk accordingly
Creating something new

- create new MODULE in rpc_server/config.mk
- declare rpc server init function in rpc_server/dcerpc_server.c
- create server directory/file in rpc_server and add actual function implementation

You're done!
How about LWISO environment?

- create idl file
- create header file defining types and data structures used
- create directory for client library – there will be two libs built: DCE/RPC stub (result of compiling idl-generated source) and client implementation
- optionally create directory for test and link the test exec against client library
Creating something new

- create directory for server – there will be (server) stub library and server exec binary implementing the actual functions
- you may need to have separate idl files for client and server

That's it!
Simple client and server

Function *LwisoCopyUniString()*
- takes 2-byte unicode string $S$ and
- unsigned integer $N$
- returns array of $N$ copies of $S$
```
#include "idl_types.h"

[  uuid("83058420-2a7d-11de-9102-001a6bd01d81"),
    version(1.0),
    endpoint("ncacn_np:\pipe\lwiso", "ncacn_ip_tcp:"),
    pointer_default(unique),
    helpstring("Likewise Open RPC test")
]
interface lwiso
{
    typedef [public] struct {
        [string, charset(UTF16)] uint16 *str;
    } lwiso_UniStr;

typedef struct {
    uint32 count;
    [size_is(count)] lwiso_UniStr *lwiso_str;
} lwiso_UniStrArray;

[public]NTSTATUS lwiso_CopyUniString(
    [in] uint32 num_copies,
    [in,string, charset(UTF16)] uint16 *str,
    [out, ref] lwiso_UniStrArray *array
);
```
Simple client and server
LWISO implementation

```plaintext
[uuid(83058420-2a7d-11de-9102-001a6bd01d81),
version(1.0),
pointer_default(unique)]

interface lwiso
{
  cpp_quote("#ifdef DCERPC_STUB_BUILD")

  #include <lwiso/lwdefs.h>

  cpp_quote("#endif")

  NTSTATUS _LwisoCopyUniString(
      [in] UINT32 uiNumCopies,
      [in,string] wchar16_t *pwszStr,
      [out,ref] LwisoUniStrArray *pArray
    );

}
```
#include <lw/types.h>
#include <lw/ntstatus.h>

typedef struct {
    #ifdef _DCE_IDL_
        [string]
    #endif
    wchar16_t *str;
} LwisoUniStr, LWISO_UNISTR, *PLWISO_UNISTR;

typedef struct {
    UINT32 uiCount;
    #ifdef _DCE_IDL_
        [size_is(uiCount)]
    #endif
    LwisoUniStr *pStr;
} LwisoUniStrArray, LWISO_UNISTR_ARRAY, *PLWISO_UNISTR_ARRAY;
Let's call each other!

LwisoCopyUniString

smbtorture: LWIS-COPY-UNISTR test
testlwiso: LWISO-COPY-UNISTR test
Conclusions

• Creating new rpc interface in Samba is a really quick process
• Building it **inside** Samba source tree is the price to be paid
• LWISO enables building a completely separate product (where you have to take care of many basic things yourself)
• Obviously it requires necessary shared libraries and headers
Sources

Likewise Open git tree:
  git://git.likewise.com/likewise-open.git

My samba git tree:
  git://git.samba.org/mimir/samba.git

Thank you for your attention!