Testing MSRPC

Andrew Tridgell
Samba Team

The other half of CIFS

- MSRPC, the variant of DCE/RPC used by Microsoft, plays a huge role in CIFS
  - the basis of almost all non-file oriented operations
  - used for resource management, user administration, directory replication, logon, printing and even file-system search

IDL and NDR

- A DCE/RPC implementation is based around two major components
    - IDL, the interface definition language, defines the structures and calls available within each interface
    - NDR, the network data representation, defines the way that a structure is linearised onto a wire buffer
- DCE/RPC without IDL?
    - in Samba3 we implemented NDR without IDL
    - in hindsight this was a mistake, it led to a very poorly structured implementation

Interpreting IDL

- IDL gives shape, but not meaning
    - a RPC test infrastructure gives the opportunity to experiment
- Here is some typical IDL
    - what does "resume_handle" contain?
    - what are the units of "max_size" ?

```
NTSTATUS samr_EnumDomainUsers(
        [in,ref]      policy_handle *handle,
        [in,out,ref]  uint32 *resume_handle,
        [in]          uint32 acct_flags,
        [in]          uint32 max_size,
        [out]         samr_SamArray *sam,
        [out]         uint32 num_entries
        );
```

A MSRPC development plan

- For Samba4 we developed our MSRPC implementation differently to our earlier attempts
    - First, form the IDL for the function
    - Second, write a test that confirms the IDL, and the meaning of elements
    - Third, write the server side implementation
- To help with the process we have developed a number of useful tools
    - ndrdump for working out IDL
    - scanners for investigating a pipe
    - IDL extensions for validating NDR and building tests

Wire vs API compatibility

- Should we be API compatible?
    - DCE/RPC defines both a programming API and a wire format
- For Samba4 we are aiming only for wire compatibility
    - only wire format is needed for remote interoperability
    - some aspects of API compatibility are unpalatable

MSRPC pipes

- Each IDL files defines one or more MSRPC pipes
- Pipes come in a number of different types
    - “database pipes” are the most common. They contain query, set and enumerate functions
        - SAMR, LSA, SPOOLSS, WKSSVC, DFS etc
    - “management pipes” provide DCE/RPC level management functions
        - MGMT and EPMAPPER
    - “specialised pipes”
        - NETLOGON, SKADS, DRSUAPI etc

Testing database pipes

- For database pipes like SAMR, the test strategy is
  - enumerate every existing element of every type in the database
  - for each existing element try every read (non-destructive) operation
  - create a new test element of each type
  - for each test element try every write (destructive) operation
  - delete the test elements

IDL extensions

- pidl adds a number of extensions to IDL
  - allows more well-known structures to be encoded as IDL
  - avoids some of the more tedious aspects of IDL coding
- major uses of the extensions so far:
  - encode low level DCE/RPC packet formats as IDL
  - encode security-descriptors as IDL
  - auto-initialise string encapsulation

subcontexts

- It is common for MS programmers to write elements like:
  - [in,size_is(length)] uint8 *buffer;”
- The MS application then needs to manually parse
- In pidl we can write:
  - [in,subcontext(4)] RealStructure *value;
- For an example, see sec_desc_buf in lsa.idl

Relative Pointers

- Microsoft use a “relative” pointer format on the wire for some structures. This would require manual parsing in midl.
- In pidl we can use:
    - [relative] uint32 *v;
- This tells pidl that the pointer should be encoded/decoded using a relative offset instead of a unique pointer
- For an example, see security_descriptor in misc.idl

Alignment and forced little-endian

- The NDR specification has strict rules for alignment, and assumes the endianness is set by the PDU flags
- In pidl those can be controlled using:
  - [flag(NDR_LITTLE_ENDIAN)]
  - [flag(NDR_NOALIGN)]
- For an example, see epm_towers in dcerpc.idl

Generating ethereal modules

- Ethereal is the best decoder of MSRPC available, but it can still be improved
  - We want to replace the RPC decoders in ethereal with auto-generated decoders based on IDL and pidl
  - This would make ethereal maintainence much easier
  - Invaluable for developers working on new pipes
- Tim Potter has this mostly working, stay tuned!

Ethereal and ndrdump

- Ethereal is a fantastic tool for investigating MSRPC pipes! It is even more useful when combined with ndrdump
    - capture a windows -> windows session using ethereal
    - use “Export Selected Packet Bytes” on the RPC payload
    - use ndrdump to dump the binary data using an IDL template
    - modify the IDL, then try ndrdump again.
    - Loop until exasperated or happy!

Auto-generating IDL - possible?

- An obvious question is whether it is possible to auto-generate IDL by probing a remote server
- The answer appears to be “sometimes”, and it certainly isn't easy
- the RPC-AUTOIDL test in smbtorture is a proof-of-concept of an IDL generator

RPC-AUTOIDL

- Try all-zero packets of length 0, 1, 2, 3 etc
- When RPC fault code changes, this gives base input structure size
    - For each 4 bytes, test if it is a pointer by varying the value from 0 to 1 and watching the fault code
    - When fault code changes, try expanding the packet by 0, 1, 2, 3 until fault code changes back
        - recursively process all pointer areas
- For an example, see samr_SetDsrmPassword

The “many ways” of MSRPC

- Like CIFS, MSRPC often has many ways of doing a operation
    - There are a total of 15 “change password” methods that we have found so far!
- This can be a blessing, as the redundency makes testing easier
- It also is a curse, as it makes it very hard to test properly using windows clients

Open Challenges

- Are you bored? Want an interesting challenge?
- There are two significant open questions we have run into but not solved in MSRPC:
    - How is the session key computed in lsa_SetSecret on a TCP transport?
    - what is the encapsulation use on the ci_skads pipe?
- See the following URLs:
    - http://samba.org/ftp/tridge/misc/lsakey.tgz
    - http://samba.org/ftp/tridge/misc/ci_skads.cap

Writing a new IDL file from scratch

- Assuming you are trying to implement an existing undocumented pipe:
    - get a sniff of windows to windows, with as many call types as possible
    - run RPC-SCANNER to find the number of calls
    - use RPC-MGMT to find the endpoint list and auth types
    - use ndrdump to try possible IDL formats for each call
    - write a smbtorture test for the new pipe

Questions?

- For a copy of this talk see
  http://samba.org/ftp/samba/slides/tridge_cifs04.pdf
- See http://devel.samba.org/ for information on downloading Samba4