SDC
STORAGE DEVELOPER CONFERENCE
SNIA ■ SANTA CLARA, 2016

Improving DCERPC Security

https://wiki.samba.org/index.php/DCERPC_Hardening

Stefan Metzmacher <metze@samba.org>

Samba Team / SerNet

2016-09-20

https://samba.org/~metze/presentations/2016/SDC/

Agenda

- The badlock related bugs
- Scope of the urgent changes
- What is DCERPC?
- Existing Hardening
- Remaining Problems
- Proposed Solutions
- Summary/Status
- Questions?

SambaXP Talk

- I gave a talk about Badlock and the related bugs at SambaXP 2016
  - https://samba.org/~metze/presentations/2016/SambaXP/
  - https://sambaxp.org
  - http://badlock.org
- I just give a short overview here...

CVE-2015-5370: Multiple errors in DCE-RPC code

- The first denial of service problem was found at an interop event by Jouni Knuutinen from Synopsys
- Jeremy Allison did the initial research
- While reviewing the initial patches the nightmare begun
- I found new problems day after day
- About 20 problem classes (mostly denial of service and man in the middle)
- Distributed over 4 DCERPC implementations (2 servers, 2 clients)
- I analysed these problems deeply together with Günther Deschner
- At the end I had 94 patches including an almost complete DCERPC protocol verification testsuite

CVE-2016-2118: Badlock (Part 1)

- While thinking about the CVE-2015-5370 patches I thought about possible related problems
- After a while I found that the DCERPC auth_level can be downgraded and nasty things can be done with it
- My first finding was limited to clients using ncacn_ip_tcp with SAMR
- I created a man in the middle exploit that got the full AD database including all secret keys while joining a Windows DC into a Windows domain
- NOTE THIS IS A FULL TAKEOVER: information leak and remote code execution on all domain member computers (maybe also in trusted domains)
- The attacker only needs to be able to intercept network traffic
- I guess it's really not that unlikely that someone might find exploits for an unpatched router firmware

CVE-2016-2118: Badlock (Part 2)

- After thinking a bit more I finally realized that the problem is even worse
- It is not limited to a join of a new Windows DC
- Every login as an administrator can be used by an attacker
- It is not limited to just Windows domains, also Samba domains are affected
- The problem is a generic to DCERPC over unprotected transports like ncacn_ip_tcp or ncacn_np (without SMB signing)
- Some application layer protocols (e.g. DRSUAPI) only allow secure connections using integrity or privacy protection
- Samba was missing most of these checks which were already available on Windows

CVE-2016-2110: NTLMSSP problems

- While working on CVE-2015-5370 and CVE-2016-2118 I thought a complete audit of all protocols was required
- After a while I found that NTLMSSP flags, e.g. NTLMSSP_SIGN/SEAL can be removed by a man in the middle without noticing
- This has implications on encrypted LDAP traffic
- A bit of research revealed that Microsoft already implemented downgrade detection into NTLMSSP when using NTLMv2
- I decided to implement the same in Samba in order to improve NTLMSSP authenticated connections

CVE-2016-2111: NETLOGON problems

- While researching about CVE-2016-2110 I found Microsofts CVE-2015-0005 "NETLOGON Spoofing Vulnerability"
- The problem with this was that any domain member was able to ask the domain controller for NTLM session keys of authentication sessions of all other domain members.
- The protection mechanism relies on NTLMv2 being used only via NTLMSSP
- During the research it turned out that the problems in Samba were even worse
- Anonymous attackers could ask for the session keys
- raw NTLMv2 was allowed without NTLMSSP wrapping, which allowed downgrade attacks

CVE-2016-2112: LDAP problems

- Fixing the specific NTLMSSP based problems of CVE-2016-2110 is not enough
- The LDAP client and server also need to verify if the authentication (gensec/gssapi) backend negotiated the requested features
- This is required in order to prevent Kerberos replay attacks
- It was required to fix these things in the LDAP server as well as in our two LDAP client libraries
- At the same time we improved the consistency of behaviors especially regarding the usage of configuration options
- The default behavior of the LDAP server is much stricter than before

CVE-2016-2113: Missing TLS certificate validation

- While analyzing CVE-2016-2110 and CVE-2016-2112, I realized that we don't do any certificate validation
- This applies to all TLS based protocols like ldaps:// and ncacn_http with https://
- For ldaps:// it only applies to tools like samba-tool, ldbsearch, ldbedit and other ldb tools
- Typically, these protocols are not used, but if someone does use them they are expected to be protected
- So (as a client) we now verify the server certificates as much as we can

CVE-2016-2114: "server signing = mandatory" not enforced

- While working on CVE-2015-5370 and CVE-2016-2118 I thought a complete audit of all protocols was required
- As all unprotected DCERPC transports are vulnerable to man in the middle attacks it was clear that SMB signing is important
- It turned out that we didn't require SMB signing even if we are configured with mandatory signing
- This is fixed now
- As an active directory domain controller we require signing by default now

CVE-2015-2115: SMB IPC traffic is not integrity protected

- While working on CVE-2015-5370 and CVE-2016-2118 I thought a complete audit of all protocols was required
- As all unprotected DCERPC transports are vulnerable to man in the middle attacks it was clear that SMB signing is important
- We can't change the default of "client signing" and "client max protocol" in a security release, because of performance reasons
- We try to use SMB3 and required signing for IPC$ related SMB client connections, which are used as a DCERPC transport

Scope of the urgent changes

- In order to prevent the man in the middle attacks it was required to change the (default) behavior for some protocols.

- As the Samba Team we only have resources to provide security fixes for 3 maintained branches (at the time 4.4, 4.3 and 4.2)
  - 4.4.2 had 323 patches on top of 4.4.0 (note that 4.4.1 had a regression and was superseeded by 4.4.2)
  - samba-4.4.0-security-2016-04-12-final.patch
    227 files changed, 14582 insertions(+), 5037 deletions(-)
  - 4.3.8 had 352 patches on top of 4.3.6 (note that 4.3.7 had a regression and was superseeded by 4.3.8)
  - samba-4.3.6-security-2016-04-12-final.patch
    236 files changed, 14870 insertions(+), 5195 deletions(-)
  - 4.2.11 had 440 patches on top of 4.2.9 (note that 4.2.10 had a regression and was superseeded by 4.2.11)
  - samba-4.2.9-security-2016-04-12-final.patch
    319 files changed, 17636 insertions(+), 7506 deletions(-)

What is DCE-RPC?

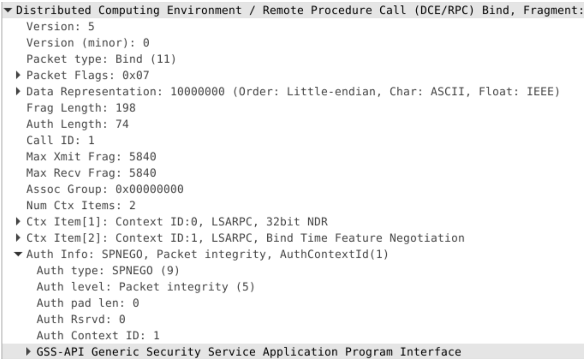
- Distributed Computing Environment / Remote Procedure Calls
  - It is an infrastructure to call a function on a remote server
  - "remote" is connected via some kind of socket (tcp/ip, named pipes, ...)

- As development environment
  - Function stubs are typically autogenerated from an Interface Definition Language (IDL)

- As network protocol defines how:
  - marshalling of payloads work - transfer syntax (NDR/NDR64)
  - marshalling of PDUs
  - PDUs are ordered
  - authentication and encryption works

- My talk from 2014 has much more details
  - https://samba.org/˜metze/presentations/2014/

Wireshark DCERPC (BIND)

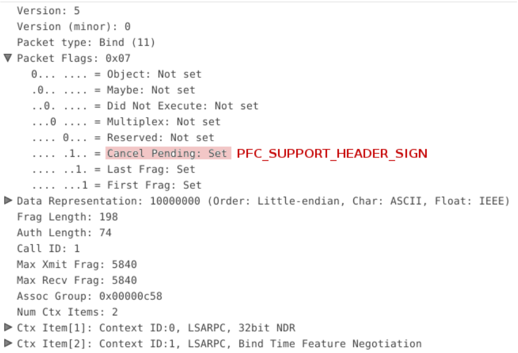


```
▼ Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Bind, Fragment:
    Version: 5
    Version (minor): 0
    Packet type: Bind (11)
  ▶ Packet Flags: 0x07
  ▶ Data Representation: 10000000 (Order: Little-endian, Char: ASCII, Float: IEEE)
    Frag Length: 198
    Auth Length: 74
    Call ID: 1
    Max Xmit Frag: 5840
    Max Recv Frag: 5840
    Assoc Group: 0x00000000
    Num Ctx Items: 2
  ▶ Ctx Item[1]: Context ID:0, LSARPC, 32bit NDR
  ▶ Ctx Item[2]: Context ID:1, LSARPC, Bind Time Feature Negotiation
  ▼ Auth Info: SPNEGO, Packet integrity, AuthContextId(1)
      Auth type: SPNEGO (9)
      Auth level: Packet integrity (5)
      Auth pad len: 0
      Auth Rsrvd: 0
      Auth Context ID: 1
    ▶ GSS-API Generic Security Service Application Program Interface
```

Existing DCERPC Hardening (PFC_SUPPORT_HEADER_SIGN)

- GSS-API based authentication is used
  - NTLMSSP, KRB5, SPNEGO
  - A custom security provider for the NETLOGON service
  - gss_wrap_iov() is required to support header signing
- MS-RPCE 2.2.2.3 PFC_SUPPORT_HEADER_SIGN Flag.
  - Same value as PFC_PENDING_CANCEL
  - This flag can be negotiated in the Bind/BindAck exchange
  - On Windows and modern Samba installations all security providers support it.
  - It protects the header fields of DCERPC Request/Response PDUs incl. the sec_trailer.

Wireshark DCERPC PFC_SUPPORT_HEADER_SIGN



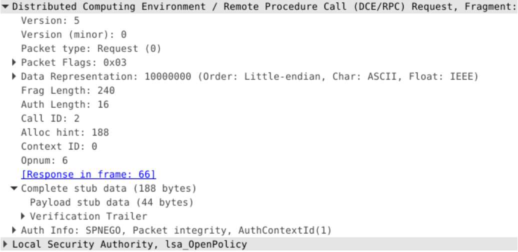
```
  Version: 5
  Version (minor): 0
  Packet type: Bind (11)
▼ Packet Flags: 0x07
    0... .... = Object: Not set
    .0.. .... = Maybe: Not set
    ..0. .... = Did Not Execute: Not set
    ...0 .... = Multiplex: Not set
    .... 0... = Reserved: Not set
    .... .1.. = Cancel Pending: Set PFC_SUPPORT_HEADER_SIGN
    .... ..1. = Last Frag: Set
    .... ...1 = First Frag: Set
▶ Data Representation: 10000000 (Order: Little-endian, Char: ASCII, Float: IEEE)
  Frag Length: 198
  Auth Length: 74
  Call ID: 1
  Max Xmit Frag: 5840
  Max Recv Frag: 5840
  Assoc Group: 0x00000c58
  Num Ctx Items: 2
▶ Ctx Item[1]: Context ID:0, LSARPC, 32bit NDR
▶ Ctx Item[2]: Context ID:1, LSARPC, Bind Time Feature Negotiation
```

Existing DCERPC hardening (Verification Trailer)

- MS-RPCE 2.2.2.13 Verification Trailer
  - A hidden structure injected at the end of the DCERPC Request stub data
  - Identified by a 8 byte magic value (0x8a, 0xe3, 0x13, 0x71, 0x02, 0xf4, 0x36, 0x71)
  - It contains an array of optional command structures

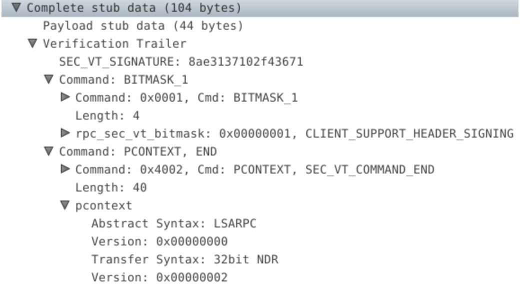
- rpc_sec_vt_bitmask protects the PFC_SUPPORT_HEADER_SIGN negotiation
- rpc_sec_vt_header2 protects the header fields if PFC_SUPPORT_HEADER_SIGN is not available
- rpc_sec_vt_pcontext protects the negotiation of the presentation context (InterfaceId/TransferSyntax)

Wireshark DCERPC Request PDU



```
▼ Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Request, Fragment:
    Version: 5
    Version (minor): 0
    Packet type: Request (0)
  ▶ Packet Flags: 0x03
  ▶ Data Representation: 10000000 (Order: Little-endian, Char: ASCII, Float: IEEE)
    Frag Length: 240
    Auth Length: 16
    Call ID: 2
    Alloc hint: 188
    Context ID: 0
    Opnum: 6
    [Response in frame: 66]
  ▼ Complete stub data (188 bytes)
      Payload stub data (44 bytes)
    ▶ Verification Trailer
  ▶ Auth Info: SPNEGO, Packet integrity, AuthContextId(1)
▶ Local Security Authority, lsa_OpenPolicy
```

Wireshark DCERPC Verification Trailer

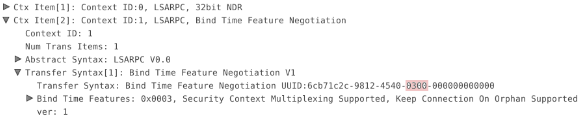


```
▼ Complete stub data (104 bytes)
    Payload stub data (44 bytes)
  ▼ Verification Trailer
      SEC_VT_SIGNATURE: 8ae3137102f43671
    ▼ Command: BITMASK_1
      ▶ Command: 0x0001, Cmd: BITMASK_1
        Length: 4
      ▶ rpc_sec_vt_bitmask: 0x00000001, CLIENT_SUPPORT_HEADER_SIGNING
    ▼ Command: PCONTEXT, END
      ▶ Command: 0x4002, Cmd: PCONTEXT, SEC_VT_COMMAND_END
        Length: 40
      ▼ pcontext
          Abstract Syntax: LSARPC
          Version: 0x00000000
          Transfer Syntax: 32bit NDR
          Version: 0x00000002
```

Existing DCERPC hardening (Bind Time Features)

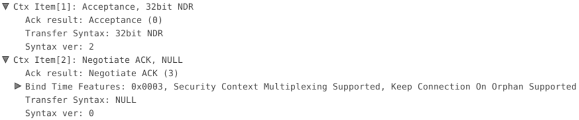
- MS-RPCE 2.2.2.14 BindTimeFeatureNegotiationBitmask
  - A way to negotiate new features
- Current defined features:
  - SecurityContextMultiplexingSupported
  - KeepConnectionOnOrphanSupported

Wireshark DCERPC Bind Time Features (BIND)



```
▶ Ctx Item[1]: Context ID:0, LSARPC, 32bit NDR
▼ Ctx Item[2]: Context ID:1, LSARPC, Bind Time Feature Negotiation
    Context ID: 1
    Num Trans Items: 1
  ▶ Abstract Syntax: LSARPC V0.0
  ▼ Transfer Syntax[1]: Bind Time Feature Negotiation V1
      Transfer Syntax: Bind Time Feature Negotiation UUID:6cb71c2c-9812-4540-0300-000000000000
    ▶ Bind Time Features: 0x0003, Security Context Multiplexing Supported, Keep Connection On Orphan Supported
      ver: 1
```

Wireshark DCERPC Bind Time Features (BIND_ACK)



```
▼ Ctx Item[1]: Acceptance, 32bit NDR
    Ack result: Acceptance (0)
    Transfer Syntax: 32bit NDR
    Syntax ver: 2
▼ Ctx Item[2]: Negotiate ACK, NULL
    Ack result: Negotiate ACK (3)
  ▶ Bind Time Features: 0x0003, Security Context Multiplexing Supported, Keep Connection On Orphan Supported
    Transfer Syntax: NULL
    Syntax ver: 0
```

Design problems of current DCERPC implementations

- DCERPC Fault, Cancel and Orphan PDUs don't include any integrity nor privacy protection.
- DCERPC_NCA_S_OP_RNG_ERROR is typically used to indicate that a specific opnum is not implemented by the server
- DCERPC_NCA_S_FAULT_INVALID_TAG is typically used to indicate that a specific information level is not supported
- There are higher level protection against downgrades required.
- The most important protocols don't have known downgrade problems.
- But it would be good to have real protection at the DCERPC layer.

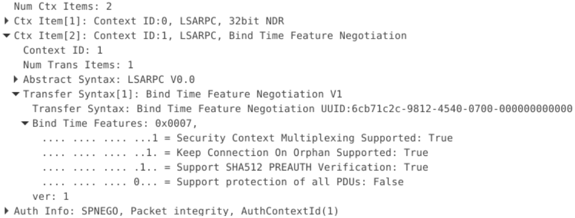
Proposed Solutions

- SMB 3.x has support for generic encryption and downgrade detection
  - It wrapps SMB 2/3 PDUs inside an SMB2 TRANSFORM_HEADER PDU.
  - FSCTL_VALIDATE_NEGOTIATE_INFO was a nice try, but does not protect everything.
- SMB 3.1.1 has finally a working downgrade protection
  - A SHA512 preauth hash is calculated over the Negotiate and SessionSetup PDUs.
- BindTimeFeatureNegotiation and Verification Trailer should be able to build a backward compatible solution for DCERPC.
  - DCERPC_BIND_TIME_SUPPORT_PREAUTH
  - DCERPC_BIND_TIME_PROTECT_ALL_PDUS
  - DCERPC_BIND_TIME_SUPPORT_WRAP

DCERPC_BIND_TIME_SUPPORT_PREAUTH

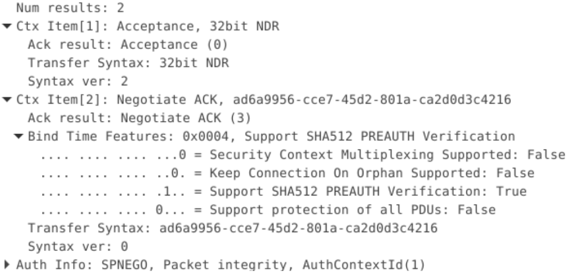
- DCERPC_BIND_TIME_SUPPORT_PREAUTH is negotiated in the Bind/BindAck exchange.
  - The DCERPC_BIND_ACK_RESULT_NEGOTIATE_ACK element is filled with a random transfer_syntax value as salt (16 bytes).
- All DCERPC Bind, BindAck, AlterContext, AlterContextResp and Auth3 PDUs update a rolling preauth hash.
  - These are triggered by the client and are strictly ordered.
  - Client and Server start with a zero preauth hash.
  - The preauth hash is updated when sending or receiving an unprotected PDU.
  - PREAUTH_SHA512 = SHA512(PREAUTH_SHA512, PDU).
- DCERPC_SEC_VT_COMMAND_PREAUTH is added to the verification trailer of the first request.
  - DCERPC_SEC_VT_COMMAND_PREAUTH contains a 16 byte SALT.
  - It also contains the result of SHA512(PREAUTH_SHA512 + SALT).

Wireshark DCERPC Bind Time Features (PREAUTH Bind)



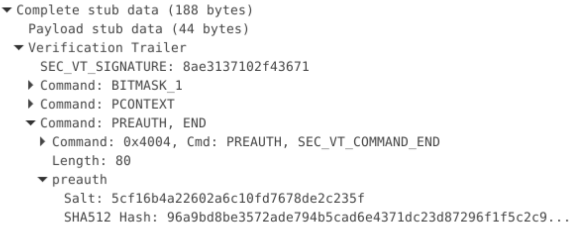
```
  Num Ctx Items: 2
▸ Ctx Item[1]: Context ID:0, LSARPC, 32bit NDR
▾ Ctx Item[2]: Context ID:1, LSARPC, Bind Time Feature Negotiation
    Context ID: 1
    Num Trans Items: 1
  ▸ Abstract Syntax: LSARPC V0.0
  ▾ Transfer Syntax[1]: Bind Time Feature Negotiation V1
      Transfer Syntax: Bind Time Feature Negotiation UUID:6cb71c2c-9812-4540-0700-000000000000
    ▾ Bind Time Features: 0x0007,
        .... .... .... ...1 = Security Context Multiplexing Supported: True
        .... .... .... ..1. = Keep Connection On Orphan Supported: True
        .... .... .... .1.. = Support SHA512 PREAUTH Verification: True
        .... .... .... 0... = Support protection of all PDUs: False
      ver: 1
▸ Auth Info: SPNEGO, Packet integrity, AuthContextId(1)
```

Wireshark DCERPC Bind Time Features (PREAUTH Ack)



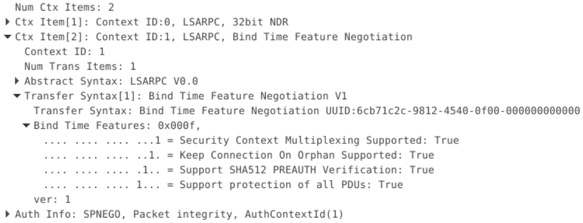
```
  Num results: 2
▾ Ctx Item[1]: Acceptance, 32bit NDR
    Ack result: Acceptance (0)
    Transfer Syntax: 32bit NDR
    Syntax ver: 2
▾ Ctx Item[2]: Negotiate ACK, ad6a9956-cce7-45d2-801a-ca2d0d3c4216
    Ack result: Negotiate ACK (3)
  ▾ Bind Time Features: 0x0004, Support SHA512 PREAUTH Verification
      .... .... .... ...0 = Security Context Multiplexing Supported: False
      .... .... .... ..0. = Keep Connection On Orphan Supported: False
      .... .... .... .1.. = Support SHA512 PREAUTH Verification: True
      .... .... .... 0... = Support protection of all PDUs: False
    Transfer Syntax: ad6a9956-cce7-45d2-801a-ca2d0d3c4216
    Syntax ver: 0
▸ Auth Info: SPNEGO, Packet integrity, AuthContextId(1)
```

Wireshark DCERPC Verification Trailer (PREAUTH)



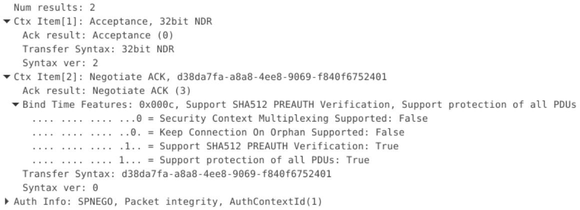
```
▼ Complete stub data (188 bytes)
    Payload stub data (44 bytes)
  ▼ Verification Trailer
      SEC_VT_SIGNATURE: 8ae3137102f43671
    ▶ Command: BITMASK_1
    ▶ Command: PCONTEXT
    ▼ Command: PREAUTH, END
      ▶ Command: 0x4004, Cmd: PREAUTH, SEC_VT_COMMAND_END
        Length: 80
      ▼ preauth
          Salt: 5cf16b4a22602a6c10fd7678de2c235f
          SHA512 Hash: 96a9bd8be3572ade794b5cad6e4371dc23d87296f1f5c2c9...
```

Wireshark DCERPC Bind Time Features (PROTECT_ALL_PDUs Bind)



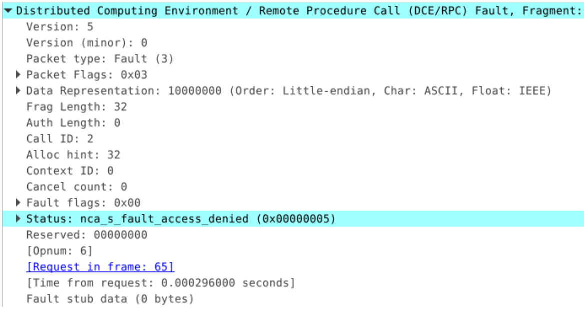
```
  Num Ctx Items: 2
▶ Ctx Item[1]: Context ID:0, LSARPC, 32bit NDR
▼ Ctx Item[2]: Context ID:1, LSARPC, Bind Time Feature Negotiation
    Context ID: 1
    Num Trans Items: 1
  ▶ Abstract Syntax: LSARPC V0.0
  ▼ Transfer Syntax[1]: Bind Time Feature Negotiation V1
      Transfer Syntax: Bind Time Feature Negotiation UUID:6cb71c2c-9812-4540-0f00-000000000000
    ▼ Bind Time Features: 0x000f,
        .... .... .... ...1 = Security Context Multiplexing Supported: True
        .... .... .... ..1. = Keep Connection On Orphan Supported: True
        .... .... .... .1.. = Support SHA512 PREAUTH Verification: True
        .... .... .... 1... = Support protection of all PDUs: True
      ver: 1
▶ Auth Info: SPNEGO, Packet integrity, AuthContextId(1)
```

Wireshark DCERPC Bind Time Features (PROTECT_ALL_PDUs Ack)



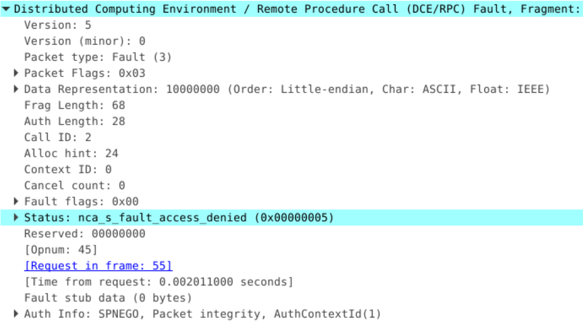
```
  Num results: 2
▼ Ctx Item[1]: Acceptance, 32bit NDR
    Ack result: Acceptance (0)
    Transfer Syntax: 32bit NDR
    Syntax ver: 2
▼ Ctx Item[2]: Negotiate ACK, d38da7fa-a8a8-4ee8-9069-f840f6752401
    Ack result: Negotiate ACK (3)
  ▼ Bind Time Features: 0x000c, Support SHA512 PREAUTH Verification, Support protection of all PDUs
      .... .... .... ...0 = Security Context Multiplexing Supported: False
      .... .... .... ..0. = Keep Connection On Orphan Supported: False
      .... .... .... .1.. = Support SHA512 PREAUTH Verification: True
      .... .... .... 1... = Support protection of all PDUs: True
    Transfer Syntax: d38da7fa-a8a8-4ee8-9069-f840f6752401
    Syntax ver: 0
▶ Auth Info: SPNEGO, Packet integrity, AuthContextId(1)
```

Wireshark DCERPC Fault PDU



```
▼ Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Fault, Fragment:
    Version: 5
    Version (minor): 0
    Packet type: Fault (3)
  ▶ Packet Flags: 0x03
  ▶ Data Representation: 10000000 (Order: Little-endian, Char: ASCII, Float: IEEE)
    Frag Length: 32
    Auth Length: 0
    Call ID: 2
    Alloc hint: 32
    Context ID: 0
    Cancel count: 0
  ▶ Fault flags: 0x00
  ▶ Status: nca_s_fault_access_denied (0x00000005)
    Reserved: 00000000
    [Opnum: 6]
    [Request in frame: 65]
    [Time from request: 0.000296000 seconds]
    Fault stub data (0 bytes)
```

Wireshark DCERPC Fault PDU (Protected)



```
▼ Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Fault, Fragment:
    Version: 5
    Version (minor): 0
    Packet type: Fault (3)
  ▶ Packet Flags: 0x03
  ▶ Data Representation: 10000000 (Order: Little-endian, Char: ASCII, Float: IEEE)
    Frag Length: 68
    Auth Length: 28
    Call ID: 2
    Alloc hint: 24
    Context ID: 0
    Cancel count: 0
  ▶ Fault flags: 0x00
  ▶ Status: nca_s_fault_access_denied (0x00000005)
    Reserved: 00000000
    [Opnum: 45]
    [Request in frame: 55]
    [Time from request: 0.002011000 seconds]
    Fault stub data (0 bytes)
  ▶ Auth Info: SPNEGO, Packet integrity, AuthContextId(1)
```

IDL definition the DCERPC (ncacn) PDU

The ncacn pdu IDL description in Samba:

```
    typedef [public] struct {
        uint8 rpc_vers;             /* RPC version */
        uint8 rpc_vers_minor;       /* Minor version */
        dcerpc_pkt_type ptype;      /* Packet type */
        dcerpc_pfc_flags pfc_flags; /* Fragmentation flags */
        uint8 drep[4];              /* NDR data representation */
        uint16 frag_length;         /* Total length of fragment */
        uint16 auth_length;         /* authenticator length */
        uint32 call_id;             /* Call identifier */
        [switch_is(ptype)] dcerpc_payload u;
    } ncacn_packet;
```

IDL definiation of the Payload union

The ncacn payload destription union:

```
    typedef [nodiscriminant] union {
        [case(DCERPC_PKT_REQUEST)]     dcerpc_request   request;
        [case(DCERPC_PKT_RESPONSE)]    dcerpc_response  response;
        [case(DCERPC_PKT_FAULT)]       dcerpc_fault     fault;
        [case(DCERPC_PKT_BIND)]        dcerpc_bind      bind;
        [case(DCERPC_PKT_BIND_ACK)]    dcerpc_bind_ack  bind_ack;
        [case(DCERPC_PKT_BIND_NAK)]    dcerpc_bind_nak  bind_nak;
        [case(DCERPC_PKT_ALTER)]       dcerpc_bind      alter;
        [case(DCERPC_PKT_ALTER_RESP)] dcerpc_bind_ack  alter_resp;
        [case(DCERPC_PKT_SHUTDOWN)]    dcerpc_shutdown  shutdown;
        [case(DCERPC_PKT_CO_CANCEL)]   dcerpc_co_cancel co_cancel;
        [case(DCERPC_PKT_ORPHANED)]    dcerpc_orphaned  orphaned;
        [case(DCERPC_PKT_AUTH3)]       dcerpc_auth3     auth3;
        [case(DCERPC_PKT_RTS)]         dcerpc_rts       rts;
        /* WRAP packets used to improve privacy */
        [case(DCERPC_PKT_WRAP)]        dcerpc_wrap      wrap;
    } dcerpc_payload;
```

dcerpc_wrap (work in progress) definition

The IDL function definition (in Samba):

```
    typedef [public] struct {
        //TODO/DISCUSS:
        //  - add random confounder at the beginning
        //  - add explicit verification traller
        //  - allow extra preauth hash check PDU
        //  - callid random?
        //  - flags?
        //  - How to detect downgrades on the client
        //    without breaking against old servers

        /* this contains the real ncacn_packet blob and the auth verifier */
        [flag(NDR_REMAINING)] DATA_BLOB pdu_and_verifier;
    } dcerpc_wrap;
```

Defining the Numbers...

- The specific numbers for flags and types need to agreed on
  - It would be good if Microsoft could assign them in MS-RPCE
  - Are other vendors also interested to implement (at least parts of) this?
- Bind Time Features:
  - DCERPC_BIND_TIME_SUPPORT_PREAUTH = 0x0004
  - DCERPC_BIND_TIME_PROTECT_ALL_PDUS = 0x0008
  - DCERPC_BIND_TIME_SUPPORT_WRAP = 0x0010
- Verification Trailer Command:
  - DCERPC_SEC_VT_COMMAND_PREAUTH = 0x0004
- PDU Type:
  - DCERPC_PKT_WRAP = 21

New DCERPC testing infrastructure

- Low-level protocol testing
  - python/samba/tests/dcerpc/raw_protocol.py
  - This uses our python bindings to marshall PDUs and use raw sockets
  - This becomes a full DCERPC testsuite exploring almost each bit in the protocol
  - Windows 2012R2 is the current reference implementation
  - Samba as AD DC also passes
  - Currently 75 tests in master and 50 more waiting for review

Calling the raw protocol testsuite (in a Samba source tree):

```
$ export SMB_CONF_PATH=/dev/null
$ export SERVER=w2012r2-188.w2012r2-l6.base
$ export USERNAME=administrator
$ export PASSWORD=A1b2C3d4
$ python/samba/tests/dcerpc/raw_protocol.py -v -f TestDCERPC_BIND
```

Application level problems (LSA and SAMR)

- Some LSA and SAMR functions use an SMB application session key
  - This implies that they only work on ncacn_np
  - They can't use DCERPC level authentication (integrity/privacy)
  - They rely on SMB signing/encryption
- There're used to be a wellknown transport session key for authenticated DCERPC
  - It was the constant ”SystemLibraryDTC”
  - All recent versions of Samba and Windows return NT_STATUS_NO_USER_SESSION_KEY instead
  - DCERPC_AUTH_LEVEL_CONNECT is not supported anymore
- samr_Connect5() and lsa_OpenPolicy2() can be used to negotiate a new behaviour
  - It's possible to avoid application level encryption
  - It could rely on DCERPC_AUTH_LEVEL_PRIVACY
  - I need to continue the discussion with Microsoft about that

Summary/Status

- DCERPC_BIND_TIME_SUPPORT_PREAUTH
  - The code is ready to be merged in to Samba master
  - Just needs some more tests
- DCERPC_BIND_TIME_PROTECT_ALL_PDUS
  - The code is ready to be merged in to Samba master
  - Just needs some more tests
- DCERPC_BIND_TIME_SUPPORT_WRAP
  - Needs a bit more thinking to get the design robust
  - There's some work in progress prototype
- The LSA and SAMR improvements
  - They need more discussion

Questions?

https://wiki.samba.org/index.php/DCERPC_Hardening

- Please contact me if you're a vendor and are interested in implementing this in your product.

- Stefan Metzmacher, `metze@samba.org`
- http://www.sernet.com

→ SerNet sponsor booth