Fachhochschule Köln
Fachgebiet Datennetze
Prof. Dr. Grebe

Studienarbeit

Active Directory Replikation

Michael Kohlgraf
michael.kohlgraf@gmx.de

Stefan Metzmacher
metze@samba.org

Wintersemester 2004/2005

Datum: 14.02.2005

Inhaltsverzeichnis:

1 Einführung:

1.1 Was ist ActiveDirectory?

ActiveDirectory [MS_W2K3_ADS] ist der Verzeichnisdienst von Windows 2000 und Windows 2003.
Ein Verzeichnisdienst ist eine verteilte Datenbank, die auf hierarchische Art und Weise Informationen über verschiedene Objekte, wie z.B. Benutzer, Gruppen oder Computer, speichert.

Diese Datenbank ist vom X.500-Informationsmodell abgeleitet und wird von einem ActiveDirectory Domänen Controller über verschiedene Protokolle den ActiveDirectory Clients zur Verfügung gestellt.

1.2 ActiveDirectory Kommunikationsprotokolle

1.2.1 Was ist LDAP ?

LDAP [LDAP_SPECS] ist ein Protokoll zum Durchsuchen und Manipulieren einer Verzeichnis Datenbank.

Das ActiveDirectory verwendet LDAP (Lightweight Directory Access Protokoll) als sein Hauptkommunikationsprotokoll.

1.2.2 Was ist Kerberos V (KRB5) ?

Kerberos [KRB5_RFC] ist ein auf geteilten Geheimnissen basierendes Authentifikationsprotokoll.

Der Key Distribution Center (KDC), der auch umgangssprachlich als Kerberos- Server genannt wird, bedient sich der ActiveDirectory Datenbank zum Speichern der Benutzer-/Passwortinformationen.

1.2.3 Was ist DNS (DDNS) ?

DNS [DNS_SPECS] ist ein Protokoll zur Übersetzung von Namen in IP-Adressen.

ActiveDirectory Domänen Controller betreiben in der Regel einen DNS (DDNS) Server, welcher die ActiveDirectory-Datenbank zur Speicherung der DNS-Daten benutzt.

ActiveDirectory-Clients benutzen das DNS Protokoll zur Namensauflösung.
Außerdem benutzen sie das DDNS Protokoll zur dynamischen Registrierung ihrer IP-Adresse(n) bei dem für die Domäne zuständigen DNS Server.

1.2.4 Was sind SAMR und NETLOGON

Protokolle die NT4-Clients zur Gruppenverwaltung und Remoteauthentifizierung benutzen sind SAMR [SAMR_IDL] bzw. NETLOGON [NETLOGON_IDL].

1.3 Was ist ActiveDirectory Replikation?

Zur verbesserten Verfügbarkeit wird ein Multimaster-Replikationsmodell benutzt, d.h. auf jedem Server sind Schreibzugriffe erlaubt, wobei Konflikte über einen Algorithmus gelöst werden. [MS_W2K3_TECH] [MS_W2K3_REPL]

Microsoft hat für seine Implementierung ein auf dem DCERPC Standard basierendes Protokoll gewählt, welches im folgenden Text DRSUAPI (Directory Service Update API) genannt wird.

Windows ActiveDirectory Domänen Controller verwenden dabei typischerweise KRB5 als Authentifizierungsprotokoll.

Außerdem erlaubt ein Windows Server nur verschlüsselte und signierte DCERPC-Verbindungen für dieses Protokoll.

2 Aufgabenstellung in dieser Studienarbeit

Die Aufgabenstellung besteht aus zwei Teilen.

Der erste Teil ist die Erstellung einer IDL-Beschreibung für die Funktionen des ActiveDirectory Replikations Dienstes. Aus Zeitgründen werde wir nur einige Funktionen auswerten und exemplarisch vorführen. Diese Funktionen könnten sein: DsBind, DsUnbind, DsCrackNames, DsGetDomainControllerInfo

Der zweite Teil besteht aus der Erstellung eines Testprogramms auf der Basis der Samba4 RPC-Infrastruktur, welches die ermittelten Funktionen als Client an einem Windows Domänen Controller testet.

3 Technisches Lösungskonzept (allgemein)

3.1 Was sind DCERPC, NDR und IDL?

3.1.1 Was ist DCERPC?

DCERPC (Distributed Computing Environment Remote Procedure Calls) [DCERPC_SPEC] ist ein Protokoll für „Entfernte Funktionsaufrufe" (Remote Procedure Call - RPC).

3.1.2 Was ist NDR?

NDR (Network Data Representation) [NDR_SPEC] ist eine Netzwerkübertragungs-Syntax, um Daten architekturunabhängig übertragen zu können.

3.1.3 Was ist IDL?

IDL (Interface Definition Language) [IDL_SPEC] ist eine Beschreibungssprache, mit der man Schnittstellen, Funktionen und strukturierte Datentypen beschreiben kann.

3.1.4 Was ist ein IDL-Compiler?

Ein IDL-Compiler (z.B. MIDL (Microsoft) oder PIDL (Samba4)) ist ein Compiler, der aus der IDL-Beschreibung Funktionen und Datentypen für eine Programmiersprache (z.B. C, C++ oder Java) generiert, welcher für die Konvertierung von Programmiersprachen-Datentypen zum (bzw. vom) NDR Format vornimmt.

3.2 Methoden zum Erarbeiten einer IDL-Beschreibung eines nicht dokumentierten RPC's

3.2.1 „Mithören" eines RPC's auf dem Netz

Im einfachsten Fall kann man die Kommunikation zwischen zwei Rechnern, die das Protokoll bereits beherrschen, mit Hilfe eines Netzwerksniffers (tcpdump, ethereal, netmon....) aufnehmen und mit einem Protokollanalyser (ethereal, netmon) betrachten. Alle Protokolle bis hin zum DCERPC sollten dort schon richtig interpretiert werden.

Ziel ist es die Klartext-Payload des DCERPC-Request/Response in eine Datei zu schreiben.

3.2.2 Was ist wenn die Daten verschlüsselt übertragen werden?

Wenn die RPC-Payload verschlüsselt übertragen wird, hilft das betrachten auf dem Netz nichts.

3.2.2.1 DCERPC-Proxy

Bei verschlüsselten Daten hilft nur ein Pseudo-RPC-Server der jede Anfrage mit einer Fehlernachricht beantwortet, oder ein RPC-Proxy der die Payload entschlüsselt und wiederverschlüsselt an einen anderen Server weiterleitet. Dabei wird die empfangene Payload in eine Datei gespeichert.

3.2.3 Was ist wenn die RPC's nicht manuell auslösbar sind?

Wenn kein Client bekannt ist, der einen bestimmten RPC auslöst, sind die oben genannten „passiven" Methoden nicht anwendbar.

3.2.3.1 Generieren von RPC's und Analyse des Fehler-Codes

In solch einem Fall kann man versuchen das Format für den RPC „aktiv" heraus zu bekommen.
Dazu kann man in einer Schleife die Payload jeweils um 1Byte, 2 Byte oder 4 Byte Schritten vergrößern und anhand der RPC-Fault-Codes die Syntax erarbeiten.

3.2.4 Analyse der Daten anhand von bekannten Grunddatentypen

Mit einem Hex-Editior kann man dann die Daten betrachten und anhand von Grunddatentypen [u]int[8|16|32|64], NTTIME, GUID, string und zusammengesetzten Typen wie struct, union, array, unique Pointer eine IDL-Beschreibung erstellen und mit einem Tool „ndrdump“ überprüfen.

3.2.5 Nützliche Helfertools

Für unsere Studienarbeit stehen uns ein paar, vom Samba-Team geschriebene, Protokollanalyse-Tools zur Verfügung, welche das Finden und Überprüfen der syntaktischen IDL-Beschreibung vereinfachen.

3.3 Erarbeitungskonzept für die Semantik der Funktionen und Daten

Hat man einmal eine Syntaktisch korrekte IDL-Beschreibung, muss man nur noch die Bedeutung der Funktion Parameter und Rückgabewerte erarbeiten.

3.3.1 Benutzung der Samba4 „smbtorture“ Infrastruktur

Die Samba4 „smbtorture“-Infrastruktur bietet eine einfache Möglichkeit RPC-CALLS zu generieren und die Daten detailliert darzustellen. Dadurch kann man die Eingabeparameter einfach variieren und die Ergebnisse analysieren.

3.3.2 Gibt es Dokumentation über die Schnittstellen Semantik ?

Außerdem gibt es teilweise Dokumentationen über die Programmierschnittstellen, welche die Bedeutung der Netzdaten erahnen lassen.

3.3.3 Gibt es alternative Protokolle, welche dieselben Daten bzw. Funktionen bereitstellen?

Sind Daten bzw. Funktionen über andere Protokolle bzw. Schnittstellen verfügbar, kann man oft durch Vergleich der Daten auf Zusammenhänge schließen.

4 Erarbeitung der IDL-Beschreibung für Funktionen des ActiveDirectory Replikations Protokoll

4.1 Erarbeitung der IDL-Syntax

Zum finden der IDL-Syntax benutzen wir eine Kombination aus den in 2x genannten Methoden.

Die “Support Tools”-Kollektion von Windows 2003 bietet nützliche Tools zum generieren von RPC’s der DRSUAPI Schnittstelle (wie z.B. “repadmin.exe” oder “ldp.exe”).

Mit Hilfe von Samba4 wird ein Pseudo-DRSUAPI-Server aufgebaut der die RPC-Payload der Anfragen der Windows-Tools in Dateien schreibt, von wo sie dann weiter analysiert werden können.

Diese Anfragen werden dann mit Hilfe von “smbtorture” nachgebildet und an einen Windows 2000/2003 Domänen Controller gesendet. Wobei dann die RPC-Payload der Serverantworten zur weiteren Analyse in Dateien geschrieben wird.

4.2 Benutzung der Microsoft Dokumentation der Programmierschnittstelle für ActiveDirectory Domain Controller und Replikations Management

Ein Vergleich der API-Datentypen mit den IDL-Datentypen gibt möglicherweise Aufschluß über die Bedeutungen der Funktionen und Daten. [MSDN_DRSUAPI]

4.3 Benutzung eines LDAP-Browsers zum Durchforsten des ActiveDirectory Verzeichnisses

Mit einem LDAP-Browser kann man sich die selben Daten wahrscheinlich auch Betrachten lassen.
Von besonderem Interesse werden wohl die Attribute “replPropertyMetaData”, “repsFrom”, “repsTo”, “replUpToDateVektor”, “objectGUID” und ähnliche sein.

5 Beispiel: DsCrackNames

5.1 Was macht die Funktion DsCrackNames()

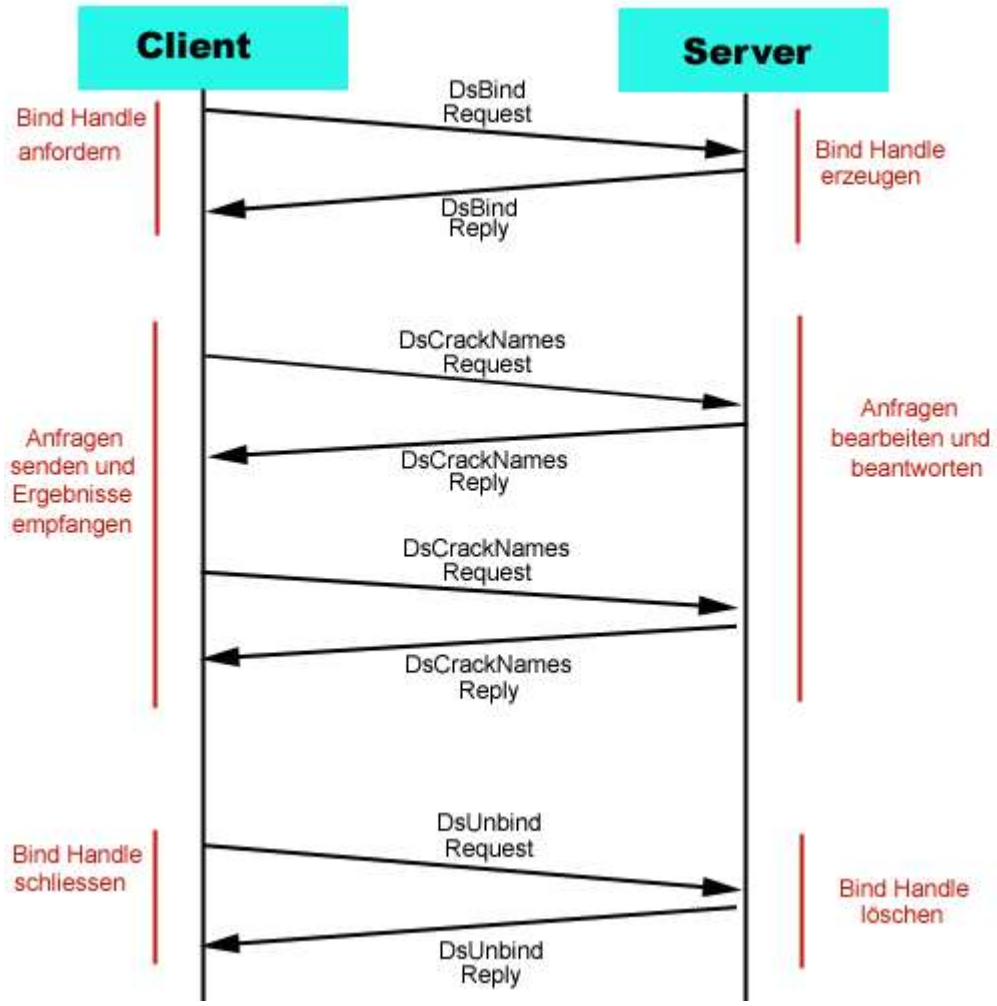
Im ActiveDirectory Verzeichnis sind Objekte über unterschiedliche Namens Typen, z.B. NT4 Namen oder LDAP DN ansprechbar. Die Funktion DsCrackNames() erledigt die Aufgabe, das eingegebene Format in das benötigte umzuwandeln. Eine genauere Beschreibung gibt es in der Microsoft API Dokumentation für diese Funktion [MSDN_DsCrackNames].

5.2 Eine kurze DRSUAPI-Session mit DsCrackNames Aufrufen

Aufgenommen mit Ethereal [ETHEREAL_WWW]

No.	Time	Source	Destination	Protocol	Info
162	22.467846	172.31.1.107	172.31.1.101	DCERPC	Bind: call_id: 1 UUID: DRSUAPI
164	22.471469	172.31.1.101	172.31.1.107	DCERPC	Bind_ack: call_id: 1 accept max_xmit: 5840 max_recv: 5840
165	22.472836	172.31.1.107	172.31.1.101	DCERPC	Alter_context: call_id: 1 UUID: DRSUAPI
166	22.473935	172.31.1.101	172.31.1.107	DCERPC	Alter_context_resp: call_id: 1 accept max_xmit: 5840 max_recv: 5840
167	22.475028	172.31.1.107	172.31.1.101	DRSUAPI	DRSBind request
168	22.476335	172.31.1.101	172.31.1.107	DRSUAPI	DRSBind response
169	22.486388	172.31.1.107	172.31.1.101	DRSUAPI	DRSCrackNames request
170	22.488084	172.31.1.101	172.31.1.107	DRSUAPI	DRSCrackNames response
171	22.488730	172.31.1.107	172.31.1.101	DRSUAPI	DRSCrackNames request
172	22.490040	172.31.1.101	172.31.1.107	DRSUAPI	DRSCrackNames response
173	22.490632	172.31.1.107	172.31.1.101	DRSUAPI	DRSUnbind request
174	22.491133	172.31.1.101	172.31.1.107	DRSUAPI	DRSUnbind response

Als Strichliniendiagramm



5.3 Verschlüsselte Verbindung bei DsCrackNames

Aufgenommen mit Ethereal [ETHEREAL_WWW]

```
⊞ Frame 169 (259 bytes on wire, 259 bytes captured)
⊞ Ethernet II, Src: 00:0c:29:11:f1:8d, Dst: 00:0c:29:09:4d:fa
⊞ Internet Protocol, Src Addr: 172.31.1.107 (172.31.1.107), Dst Addr: 172.31.1.101 (172.31.1.101)
⊞ Transmission Control Protocol, Src Port: 1060 (1060), Dst Port: 1025 (1025), Seq: 1978045736, Ack: 1297032480, Len: 205
⊟ DCE RPC
    Version: 5
    Version (minor): 0
    Packet type: Request (0)
  ⊞ Packet Flags: 0x03
  ⊞ Data Representation: 10000000
    Frag Length: 205
    Auth Length: 45
    Call ID: 2
    Alloc hint: 118
    Context ID: 0
    Opnum: 12
    Auth type: SPNEGO (9)
    Auth level: Packet privacy (6)
    Auth pad len: 10
    Auth Rsrvd: 0
    Auth Context ID: 1
    [Response in frame: 170]
  ⊞ GSS-API
⊟ Microsoft Directory Replication Service, DRSCrackNames
    Operation: DRSCrackNames (12)
    Encrypted stub data (128 bytes)
```

```
0040  2d 00 02 00 00 00 76 00  00 00 00 00 0c 00 62 54   -.....v. ......bT
0050  9c 7f b1 75 70 54 f5 19  d2 1f 29 99 cf f4 54 1d   ...upT.. ..)...T.
0060  61 e6 a6 d3 f8 c8 30 4c  d3 74 b0 95 89 12 6b fb   a.....0L .t....k.
0070  c6 50 d8 06 04 8e b6 c3  e4 a8 f7 d4 30 13 8e 33   .P...... ....0..3
0080  d9 86 83 f3 67 9e d0 c7  33 83 40 e1 ae 00 0a 72   ....g... 3.@....r
0090  0d 16 88 f4 00 1a a2 06  e7 3b a9 04 19 93 d4 a7   ........ .;......
00a0  c7 c9 f3 85 ff 74 0a 2b  d3 09 ec 94 5a ce 3b 9f   .....t.+ ....Z.;.
00b0  ee d4 05 11 4c 93 aa 3e  c7 db b0 78 c4 6c 6e ae   ....L..> ...x.ln.
00c0  d7 65 15 dd 18 02 70 39  48 c1 9d ba 90 e6 09 06   .e....p9 H.......
00d0  0a 00 01 00 00 00 60 2b  06 09 2a 86 48 86 f7 12   ......`+ ..*.H...
00e0  01 02 02 02 01 11 00 10  00 ff ff a7 ee 8e 20 9f   ........ ...... .
00f0  17 f2 79 42 7e 4c 2e ac  3d 19 56 c3 80 1b 82 7f   ..yB~L.. =.V.....
0100  bc 13 e2                                          ...
```

5.4 Smbtorture Output für DsCrackNames

#~samba4/source> bin/smbtorture ncacn_ip_tcp:w2k3-101[seal,print] -U administrator%test -W W2K3 --option="realm=w2k3.vmnet1.vm.base" RPC-DRSUAPI -d 10

```
testing DsCrackNames with name 'W2K3\W2K3-101$' desired format:1
    drsuapi_DsCrackNames: struct drsuapi_DsCrackNames
        in: struct drsuapi_DsCrackNames
            bind_handle              : *
                bind_handle: struct policy_handle
                    handle_type              : 0x00000000 (0)
                    uuid                     : 16137fd0-c8fc-434c-b1d1-df3fb3187b0c
            level                    : 1
            req                      : union drsuapi_DsNameRequest(case 1)
            req1: struct drsuapi_DsNameRequest1
                unknown1                 : 0x000004e4 (1252)
                unknown2                 : 0x00000407 (1031)
                format_flags             : DRSUAPI_DS_NAME_FLAG_NO_FLAGS (0)
                format_offered           : DRSUAPI_DS_NAME_FORMAT_NT4_ACCOUNT (2)
                format_desired           : DRSUAPI_DS_NAME_FORMAT_FQDN_1779 (1)
                count                    : 0x00000001 (1)
                names                    : *
                    names: ARRAY(1)
                        [0]: struct drsuapi_DsNameString
                            str                      : *
                                str                      : 'W2K3\W2K3-101$'
rpc request data:
[000] 00 00 00 00 D0 7F 13 16  FC C8 4C 43 B1 D1 DF 3F  ........ ..LC...?
[010] B3 18 7B 0C 01 00 00 00  01 00 00 00 E4 04 00 00  ..{..... ........
[020] 07 04 00 00 00 00 00 00  02 00 00 00 01 00 00 00  ........ ........
[030] 01 00 00 00 01 00 00 00  01 00 00 00 02 00 00 00  ........ ........
[040] 0F 00 00 00 00 00 00 00  0F 00 00 00 57 00 32 00  ........ ....W.2.
[050] 4B 00 33 00 5C 00 57 00  32 00 4B 00 33 00 2D 00  K.3.\.W. 2.K.3.-.
[060] 31 00 30 00 31 00 24 00  00 00                    1.0.1.$. ..
rpc reply data:
[000] 01 00 00 00 01 00 00 00  00 00 02 00 01 00 00 00  ........ ........
[010] 04 00 02 00 01 00 00 00  00 00 00 00 08 00 02 00  ........ ........
[020] 0C 00 02 00 14 00 00 00  00 00 00 00 14 00 00 00  ........ ........
[030] 77 00 32 00 6B 00 33 00  2E 00 76 00 6D 00 6E 00  w.2.k.3. ..v.m.n.
[040] 65 00 74 00 31 00 2E 00  76 00 6D 00 2E 00 62 00  e.t.1... v.m...b.
[050] 61 00 73 00 65 00 00 00  42 00 00 00 00 00 00 00  a.s.e... B.......
[060] 42 00 00 00 43 00 4E 00  3D 00 57 00 32 00 4B 00  B...C.N. =.W.2.K.
[070] 33 00 2D 00 31 00 30 00  31 00 2C 00 4F 00 55 00  3.-.1.0. 1.,.O.U.
[080] 3D 00 44 00 6F 00 6D 00  61 00 69 00 6E 00 20 00  =.D.o.m. a.i.n. .
[090] 43 00 6F 00 6E 00 74 00  72 00 6F 00 6C 00 6C 00  C.o.n.t. r.o.l.l.
[0A0] 65 00 72 00 73 00 2C 00  44 00 43 00 3D 00 77 00  e.r.s.,. D.C.=.w.
[0B0] 32 00 6B 00 33 00 2C 00  44 00 43 00 3D 00 76 00  2.k.3.,. D.C.=.v.
[0C0] 6D 00 6E 00 65 00 74 00  31 00 2C 00 44 00 43 00  m.n.e.t. 1.,.D.C.
[0D0] 3D 00 76 00 6D 00 2C 00  44 00 43 00 3D 00 62 00  =.v.m.,. D.C.=.b.
[0E0] 61 00 73 00 65 00 00 00  00 00 00 00              a.s.e... ....
    drsuapi_DsCrackNames: struct drsuapi_DsCrackNames
        out: struct drsuapi_DsCrackNames
            level                    : 1
            ctr                      : union drsuapi_DsNameCtr(case 1)
            ctr1                     : *
                ctr1: struct drsuapi_DsNameCtr1
                    count                    : 0x00000001 (1)
                    array                    : *
                        array: ARRAY(1)
                            [0]: struct drsuapi_DsNameInfo1
                                status                   : DRSUAPI_DS_NAME_STATUS_OK
(0)
                                dns_domain_name          : *
                                    dns_domain_name          : 'w2k3.vmnet1.vm.base'
                                result_name              : *
                                    result_name              : 'CN=W2K3-101,OU=Domain
 Controllers,DC=w2k3,DC=vmnet1,DC=vm,DC=base'
            result                   : WERR_OK
```

5.5 IDL-Beschreibung für DsCrackNames

```
00: /* GUID 16 random bytes on the wire */
01: typedef [public,noprint,gensize] struct {
02:       uint32 time_low;
03:       uint16 time_mid;
04:       uint16 time_hi_and_version;
05:       uint8  clock_seq[2];
06:       uint8  node[6];
07: } GUID;
08:
09: typedef [public] struct {
10:       uint32 handle_type;
11:       GUID   uuid;
12: } policy_handle;
13:
14: /*****************/
15: /* DsCrackNames      */
16: /*****************/
17:
18: typedef [v1_enum] enum {
19:       DRSUAPI_DS_NAME_STATUS_OK                                 = 0,
20:       DRSUAPI_DS_NAME_STATUS_RESOLVE_ERROR                      = 1,
21:       DRSUAPI_DS_NAME_STATUS_NOT_FOUND                          = 2,
22:       DRSUAPI_DS_NAME_STATUS_NOT_UNIQUE                         = 3,
23:       DRSUAPI_DS_NAME_STATUS_NO_MAPPING                         = 4,
24:       DRSUAPI_DS_NAME_STATUS_DOMAIN_ONLY                        = 5,
25:       DRSUAPI_DS_NAME_STATUS_NO_SYNTACTICAL_MAPPING             = 6,
26:       DRSUAPI_DS_NAME_STATUS_TRUST_REFERRAL                     = 7
27: } drsuapi_DsNameStatus;
28:
29: typedef [v1_enum] enum {
30:       DRSUAPI_DS_NAME_FLAG_NO_FLAGS                  = 0x0,
31:       DRSUAPI_DS_NAME_FLAG_SYNTACTICAL_ONLY          = 0x1,
32:       DRSUAPI_DS_NAME_FLAG_EVAL_AT_DC                = 0x2,
33:       DRSUAPI_DS_NAME_FLAG_GCVERIFY                  = 0x4,
34:       DRSUAPI_DS_NAME_FLAG_TRUST_REFERRAL            = 0x8
35: } drsuapi_DsNameFlags;
36:
37: typedef [v1_enum] enum {
38:       DRSUAPI_DS_NAME_FORMAT_UKNOWN                  = 0,
39:       DRSUAPI_DS_NAME_FORMAT_FQDN_1779               = 1,
40:       DRSUAPI_DS_NAME_FORMAT_NT4_ACCOUNT             = 2,
41:       DRSUAPI_DS_NAME_FORMAT_DISPLAY                 = 3,
42:       DRSUAPI_DS_NAME_FORMAT_GUID                    = 6,
43:       DRSUAPI_DS_NAME_FORMAT_CANONICAL               = 7,
44:       DRSUAPI_DS_NAME_FORMAT_USER_PRINCIPAL          = 8,
45:       DRSUAPI_DS_NAME_FORMAT_CANONICAL_EX            = 9,
46:       DRSUAPI_DS_NAME_FORMAT_SERVICE_PRINCIPAL       = 10,
47:       DRSUAPI_DS_NAME_FORMAT_SID_OR_SID_HISTORY      = 11,
48:       DRSUAPI_DS_NAME_FORMAT_DNS_DOMAIN              = 12
49: } drsuapi_DsNameFormat;
50:
51: typedef struct {
52:       unistr *str;
53: } drsuapi_DsNameString;
```

```
54: typedef struct {
55:       uint32 unknown1; /* 0x000004e4 */
56:       uint32 unknown2; /* 0x00000407 */
57:       drsuapi_DsNameFlags format_flags;
58:       drsuapi_DsNameFormat format_offered;
59:       drsuapi_DsNameFormat format_desired;
60:       [range(1,10000)] uint32 count;
61:       [size_is(count)] drsuapi_DsNameString *names;
62: } drsuapi_DsNameRequest1;
63:
64: typedef union {
65:       [case(1)] drsuapi_DsNameRequest1 req1;
66: } drsuapi_DsNameRequest;
67:
68: typedef struct {
69:       drsuapi_DsNameStatus status;
70:       unistr *dns_domain_name;
71:       unistr *result_name;
72: } drsuapi_DsNameInfo1;
73:
74: typedef struct {
75:       uint32 count;
76:       [size_is(count)] drsuapi_DsNameInfo1 *array;
77: } drsuapi_DsNameCtr1;
78:
79: typedef union {
80:       [case(1)] drsuapi_DsNameCtr1 *ctr1;
81: } drsuapi_DsNameCtr;
82:
83: /* Function 0x0C*/
84: WERROR drsuapi_DsCrackNames(
85:       [in,ref] policy_handle *bind_handle,
86:       [in, out] int32 level,
87:       [in,switch_is(level)] drsuapi_DsNameRequest req,
88:       [out,switch_is(level)] drsuapi_DsNameCtr ctr
89:       );
```

5.6 Erklärungen zur IDL-Beschreibung von DsCrackNames

5.6.1 DsCrackNames Request

Unter 5.4 ist die DCRPC Request Payload unter „rpc request data:“ als Hex-Dump dargestellt.
Es wird nun der Zusammenhang der einzelnen Bytes mit der IDL-Beschreibung diskutiert.

5.6.1.1 Die Bytes 0-19

Die Bytes 0-19 (0x000-0x013) sind ein standard DCERPC POLICY-HANDLE, wie es auch vielen anderen Protokollen vorkommt. In der IDL-Beschreibung findet man dieses in Zeile 85 als „bind_handle“. Die Option [in] besagt, dass der Parameter nur als Input-Parameter dient und nur im Request und nicht im Reply vorhanden ist. Die Option [ref] besagt, dass der Parameter nur in der API und nicht auf dem Netz ein Pointer ist.

Das „bind_handle“ wird von der Funktion DsBind generiert und ist auch in jeder anderen DRSUPI-Funktion der erste Input-Parameter.

5.5.1.2 Die Bytes 20-27

Die Bytes 20-23 (0x014-0x017) sind eine 32-Bit Integerzahl. Die Betrachtung der nächsten 4 Bytes (24-27) haben als 32-Bit Integerzahl den selben Wert, was darauf schließen lässt, dass es sich um eine LEVEL-Variable gefolgt von einem UNION handelt. Siehe Zeilen 86-87 und 64-66 in der IDL-Beschreibung.

5.5.1.3 Die Bytes 28-63

Die Bytes ab Byte 28 sind abhängig vom UNION-Level, welches hier immer nur „1“ ist. Microsoft hat dies wahrscheinlich eingeführt, um später andere Funktionsvarianten hinzufügen zu können, ohne das ältere Server oder Clients geändert werden müssen.

Die Bytes 28-35 (0x01C-0x023) sind bis jetzt noch unbekannt und werden daher jeweils als 32-Bit Unsigned Integer geführt. Siehe Zeilen 55-56 der IDL-Beschreibung.

Die Bytes 36-39 (0x024-0x027) sind ein 32-Bit ENUM vom Typ „DsNameFlags“.
Siehe Zeile 57 und Zeilen 29-35. Die Option [v1_enum] besagt, dass 32-Bit verwendet werden.

Die Bytes 40-47 (0x028-0x02F) sind zwei 32-Bit ENUM’s vom Typ „DsNameFormat“.
Siehe Zeile 57 und Zeilen 29-35. Die Option [v1_enum] besagt, dass 32-Bit verwendet werden.
Die Variable „format_offered“ gibt das gegebene Namensformat an.
Die Variable „format_desired“ gibt das erwartete Namensformat an.

Die Bytes 48-51 (0x030-0x033) sind eine 32-Bit Unsigned Integer Zahl, welche die Größe des später folgenden ARRAYs angibt. Siehe Zeile 60 in der IDL-Beschreibung, die Option [range(1,10000)] besagt, dass nur Werte zwischen 1 und 10000 gültig sind.

Die Bytes 52-55 (0x034-0x037) ist ein UNIQUE-Pointer, für das folgende ARRAY. Leider zählt Samba4 die Werte der UNIQUE Pointers von 1 in 1ner-Schritten hoch, deshalb ist der Pointer hier nicht direct ersichtlich, bei Windows-Packeten sind die Pointer besser erkennbar. Siehe Zeile 61 der IDL-Beschreibung.

Ab Byte 56 (0x038) folgt nun das ARRAY „names“, welches mit 32-Bit für die Anzahl der Elemente startet (der selbe Wert wie die „count“ Variable vorher!

Ab Byte 60 (0x03C) folgen die ARRAY-Elemente, hier vom Typ DsNameString, welche nur einem UNIQUE Pointer auf einen „unistr“-String enthält. Wir haben hier nur eine Element und die Bytes 60-63 sind also der Pointer.
Siehe Zeile 51-53 der IDL-Beschreibung.

5.5.1.4 Die Bytes 64-...

Danach folgt der „unistr“-String, ab Byte 64 (0x040), dieser ist UTF-16 Codiert und Null-Terminiert. Außerdem hat er einen Header bestehend aus 4 Byte Size, 4 Byte Offset und 4 Byte Length, wobei Size == Length und Offset == 0 gilt. Der eigentliche String beginnt bei Byte 76 (0x04C) und beendet den DsCrackNames() Request bei nach Byte 105 (0x069).

5.5.2 DsCrackNames Reply

Unter 5.4 ist die DCRPC Reply Payload unter „rpc reply data:“ als Hex-Dump dargestellt.
Es wird nun der Zusammenhang der einzelnen Bytes mit der IDL-Beschreibung diskutiert

5.5.2.2 Die Bytes 0-7

Die Bytes 0-3 (0x000-0x003) sind wieder ein 32-Bit Level, dem in den Bytes 4-7 (0x004-0x007) das UNION-Level folgt. Siehe Zeilen 86 und 88 der IDL-Beschreibung.

5.5.2.3 Die Bytes 8-11

Die Bytes 8-11 (0x008-0x00B) sind der Pointer zum DsNameCtr1, siehe Zeilen 79-81 der IDL-Beschreibung.

5.5.2.4 Die Bytes 12-23

Die Bytes 12-15 (0x00C-0x00F) sind das 32-Bit Unsinged „count“ aus Zeile 75, die Bytes 16-19 (0x010-0x13) sind der UNIQUE-Pointer auf das DsNameInfo1 ARRAY siehe Zeile 76. Die Bytes 20-23 (0x014-0x017) enthalten noch mal die Anzahl der ARRAY-Elemente als 32-Bit Integer.

5.5.2.5 Die Bytes 24-27

Die Bytes 24-27 (0x018-0x01B) sind das DsNameStatus 32-Bit Enum von DsNameInfo1, welcher den Erfolg oder Mißerfolg der Names-Konvertierung liefert. Siehe Zeilen 18-27 und 69 der IDL-Beschreibung.

5.5.2.6 Die Bytes 28-35

Die Bytes 28-31 (0x01C-0x01F) sind der UNIQUE-Pointer auf den „dns_domain_name“ unistr, siehe Zeile 70 der IDL-Beschreibung. Die Bytes 32-35 (0x020-0x023)) sind der UNIQUE-Pointer auf den „result_name“ unistr, siehe Zeile 71 der IDL-Beschreibung.

5.5.2.7 Die Bytes 32-87

Die Bytes 32-87 (0x024-0x057) enthalten den „unistr“-String für dns_domain_name, welcher wie in 5.6.1.4 beschrieben aufgebaut ist. Dies ist der DNS-Domain-Name der ActiveDirectory Domäne aus dem der anzufragende Name stammt.

5.5.2.8 Die Bytes 88-231

Die Bytes 88-231 (0x058-0x0E7) enthalten den „unistr“-String für result_name, welcher wie in 5.6.1.4 beschrieben aufgebaut ist. Dies ist das Resultat der NamensKonvertierung, dessen Format dem des „format_desired“ Feld des DsCrackNames() Request entspricht.

5.5.2.9 Die Bytes 232-235

Die Bytes 232-235 (0x0E8-0x0EB) enthalten den WERROR Return-Code der gesamten Funktion.
Siehe Samba’s doserr.h [WERROR_H] als Referenz für mögliche Return-Codes.

6 Der „RPC-DRSUAPI“ Test

6.1 Kommandozeilenoptionen für smbtorture

```
#~/samba4/source> bin/smbtorture --help
Usage: <binding>|<unc> TEST1 TEST2 ...
        -p, --smb-ports=STRING          SMB ports
        --seed=INT                      seed
        --num-progs=INT                 num progs
        --num-ops=INT                   num ops
        --entries=INT                   entries
        -L, --use-oplocks                       use oplocks
        --show-all                      show all
        --loadfile=STRING               loadfile
        --unclist=STRING                unclist
        -t, --timelimit=STRING          timelimit
        -f, --failures=INT                      failures
        -D, --parse-dns=STRING          parse-dns
        -X, --dangerous                 dangerous

Help options:
        -?, --help                      Show this help message
        --usage                         Display brief usage message

Common samba options:
        -d, --debuglevel=DEBUGLEVEL     Set debug level
        -s, --configfile=CONFIGFILE     Use alternative configuration file
        --option=name=value             Set smb.conf option from command
                                        line
        -l, --log-basename=LOGFILEBASE          Basename for log/debug files
        --leak-report                   enable talloc leak reporting on exit
        --leak-report-full              enable full talloc leak reporting on exit

Connection options:
        -R, --name-resolve=NAME-RESOLVE-ORDER   Use these name resolution services
                                                only
        -O, --socket-options=SOCKETOPTIONS      socket options to use
        -n, --netbiosname=NETBIOSNAME           Primary netbios name
        -W, --workgroup=WORKGROUP               Set the workgroup name
        -i, --scope=SCOPE                       Use this Netbios scope
        -m, --maxprotocol=MAXPROTOCOL           Set max protocol level

Authentication options:
        -U, --user=USERNAME                     Set the network username
        -N, --no-pass                           Don't ask for a password
        -k, --kerberos                          Use kerberos (active directory)
                                                authentication
        -A, --authentication-file=FILE          Get the credentials from a file
        -S, --signing=on|off|required           Set the client signing state
        -P, --machine-pass                      Use stored machine account password

Common samba options:
        -V, --version                           Print version
```

6.2 Detaillierte Ausgabe des Testcodes bei einem Windows 2003 Server

#~/samba4/source>bin/smbtorture ncacn_ip_tcp:w2k3-101[seal,print] -U administrator%test -W W2K3
--option="realm=w2k3.vmnet1.vm.base" RPC-DRSUAPI -d 10
[DRSUAPI_TEST_OUT]

#~/samba4/source>bin/smbtorture ncacn_ip_tcp:w2k3-101[seal,print] -U administrator%test -W W2K3
--option="realm=w2k3.vmnet1.vm.base" RPC-DRSUAPI -d 10
[DRSUAPI_TEST_OUT_D10]

7 Fazit

Bis jetzt sind folgende Funktionen größtenteils in IDL beschrieben:
DsBind(), DsUnbind(), DsReplicaSync(),DsCrackNames(),DsWriteAccountSpn(), DsGetDomainControllerInfo() und DsReplicaGetInfo().

Dies bietet schon eine gute Grundlage für weitere Erforschungen, die innerhalb des Samba Projektes [SAMBA_WWW] fortgeführt wird.

Leider ist die Funktion DsGetNCChanges(), welche für die eigentliche Replikation aller Daten benötigt wird, noch nicht decodiert worden. Der Grund hierfür ist, dass Samba4 noch nicht mit GSSAPI/KRB5 verschlüsselten DCERPC-Verbindungen umgehen kann und Windows DsGetNCChanges() nur über solche Verbindungen generiert.

8 Links zum Source-Code, usw

8.1 Samba4 DRSUAPI Source-Code

IDL-Beschreibung: [DRSUAPI_IDL]

Testcode: [DRSUAPI_TEST]

Servercode: [DRSUAPI_SERVER]

8.2 Kompletter Samba4 Source-Code

Samba 4 Quellen via http:
http://websvn.samba.org/cgi-bin/viewcvs.cgi/branches/SAMBA_4_0/
oder
http://samba.org/ftp/unpacked/samba4/

Samba 4 Quellen via subversion [SUBVERSION_WWW]:
svn co svn://svnanon.samba.org/samba/branches/SAMBA_4_0 samba4

8.3 Samba4 compilieren

```
#>cd samba4/source/
#>./autogen.sh
#>./configure  --prefix=$HOME/samba4/
#>make
#>make install
```

8.4 Samba Entwicklungs-Diskussion

Die Entwicklung von Samba wird auf der „samba-technical“-Mailingliste (http://www.samba.org/samba/archives.html) und dem „#samba-technical“-IRC-Channel (http://www.samba.org/samba/irc.html) diskutiert und koordiniert.

9 Quellenverzeichnis

[LDAP_SPECS]
http://www.mozilla.org/directory/standards.html
[KRB5_RFC]
http://www.faqs.org/rfcs/rfc1510.html
[DNS_SPECS]
http://www.dns.net/dnsrd/rfc/
[SAMBA_WWW]
http://www.samba.org/
[SAMR_IDL]
http://websvn.samba.org/cgi-bin/viewcvs.cgi/branches/SAMBA_4_0/source/librpc/idl/samr.idl?view=markup
[NETLOGON_IDL]
http://websvn.samba.org/cgi-bin/viewcvs.cgi/branches/SAMBA_4_0/source/librpc/idl/netlogon.idl?view=markup
[DCERPC_SPEC]
http://www.opengroup.org/onlinepubs/9629399/toc.htm
[NDR_SPEC]
http://www.opengroup.org/onlinepubs/9629399/chap14.htm#tagcjh_19
[IDL_SPEC]
http://www.opengroup.org/onlinepubs/9629399/chap4.htm#tagcjh_08
[DRSUAPI_IDL]
http://websvn.samba.org/cgi-bin/viewcvs.cgi/branches/SAMBA_4_0/source/librpc/idl/drsuapi.idl?view=markup
[DRSUAPI_TEST]
http://websvn.samba.org/cgi-bin/viewcvs.cgi/branches/SAMBA_4_0/source/torture/rpc/drsuapi.c?view=markup
[DRSUAPI_SERVER]
http://websvn.samba.org/cgi-bin/viewcvs.cgi/branches/SAMBA_4_0/source/rpc_server/drsuapi/
[DRSUAPI_TEST_OUT]
http://samba.org/~metze/presentations/2005/Datennetze1/drsuapi-test-out.txt
[DRSUAPI_TEST_OUT_D10]
http://samba.org/~metze/presentations/2005/Datennetze1/drsuapi-test-out-d10.txt
[WERROR_H]
http://websvn.samba.org/cgi-bin/viewcvs.cgi/branches/SAMBA_4_0/source/include/doserr.h?view=markup
[MSDN_DsCrackNames] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ad/ad/dscracknames.asp
[MSDN_DRSUAPI]
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ad/ad/dc_and_replication_management_functions.asp
[MS_W2K3_ADS]
http://www.microsoft.com/resources/documentation/WindowsServ/2003/all/techref/en-us/W2K3TR_ad_over.asp
[MS_W2K3_TECH]
http://www.microsoft.com/resources/documentation/WindowsServ/2003/all/techref/en-us/default.asp
[MS_W2K3_REPL]
http://www.microsoft.com/resources/documentation/WindowsServ/2003/all/techref/en-us/w2k3tr_repup_how.asp
[ETHEREAL_WWW] http://www.ethereal.com/
[SUBVERSION_WWW] http://subversion.tigris.org