Kerberos working group
Internet Draft
Document: draft-brezak-win2k-krb-authz-01.txt
Category: Informational

John Brezak
Microsoft

October, 2002

Utilizing the Windows 2000 Authorization Data in Kerberos Tickets for Access Control to Resources

Status of this Memo

1. Abstract

Microsoft Windows 2000 includes operating system specific data in the Kerberos V5 [1] authorization data field that is used for access control. This data is used to create an NT access token. The access token is used by the system to enforce access checking when attempting to access objects. This document describes the structure of the Windows 2000 specific authorization data that is carried in that field for use by servers in performing access control.

2. Conventions used in this document

All defined data structures are defined using "C" style constructs unless otherwise stated. All data is encoded as little-endian.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [2].

3. Top-Level PAC Structure

The PAC is generated by the KDC under the following conditions:

- during an AS request that has been validated with pre-authentication
- during a TGS request when the client has no PAC and the target is a service in the domain or a ticket granting service (referral ticket).

The PAC itself is included in the IF-RELEVANT (ID 1) portion of the authorization data in a ticket. Within the IF-RELEVANT portion, it is encoded KERB_AUTH_DATA_PAC with ID 128.

The PAC is defined as a C data type, with integers encoded in little-endian order. The PAC itself is made up of several layers. The outer structure, contained directly in the authorization data, is as follows. The top-level structure is the PACTYPE structure:

```
typedef unsigned long ULONG;
typedef unsigned short USHORT;
typedef unsigned long64 ULONG64;
typedef unsigned char UCHAR;

typedef struct _PACTYPE {
    ULONG cBuffers;
    ULONG Version;
    PAC_INFO_BUFFER Buffers[1];
} PACTYPE;
```

The fields are defined as follows:
cBuffers - contains the number of entries in the array Buffers
Version - this is version zero
Buffers - contains a conformant array of PAC_INFO_BUFFER structures

The PAC_INFO_BUFFER structure contains information about each piece of the PAC.

```
typedef struct _PAC_INFO_BUFFER {
    ULONG ulType;
    ULONG cbBufferSize;
    ULONG64 Offset;
} PAC_INFO_BUFFER;
```

Type fields are defined as follows:

ulType - contains the type of data contained in this buffer. For Windows 2000 access control, it may be one of the following, which are explained further below:

```
#define PAC_LOGON_INFO              1
#define PAC_SERVER_CHECKSUM         6
#define PAC_PRIVSVR_CHECKSUM        7
```

Offset - contains the offset to the beginning of the data, in bytes, from the beginning of the PACTYPE structure. The data offset must by a multiple of 8. If the data pointed to by this

field is complex, the data is typically NDR encoded. If the data is simple (indicating it includes no pointer types or complex structures) it is a little-endian format data structure.

4. PAC Credential Information (PAC_LOGON_INFO)

PAC_INFO_BUFFERs of type PAC_LOGON_INFO contain the credential information for the client of the Kerberos ticket. The data itself is contained in a KERB_VALIDATION_INFO structure, which is NDR encoded. The output of the NDR encoding is placed in the PAC_INFO_BUFFER structure of type PAC_LOGON_INFO.

```
typedef struct _KERB_VALIDATION_INFO {
    FILETIME Reserved0;
    FILETIME Reserved1;
    FILETIME KickOffTime;
    FILETIME Reserved2;
    FILETIME Reserved3;
    FILETIME Reserved4;
    UNICODE_STRING Reserved5;
    UNICODE_STRING Reserved6;
    UNICODE_STRING Reserved7;
    UNICODE_STRING Reserved8;
    UNICODE_STRING Reserved9;
    UNICODE_STRING Reserved10;
    USHORT Reserved11;
    USHORT Reserved12;
    ULONG UserId;
    ULONG PrimaryGroupId;
    ULONG GroupCount;
    [size_is(GroupCount)] PGROUP_MEMBERSHIP GroupIds;
    ULONG UserFlags;
    ULONG Reserved13[4];
    UNICODE_STRING Reserved14;
    UNICODE_STRING Reserved15;
    PSID LogonDomainId;
    ULONG Reserved16[2];
```

```
          ULONG Reserved17;
          ULONG Reserved18[7];
          ULONG SidCount;
          [size_is(SidCount)] PKERB_SID_AND_ATTRIBUTES ExtraSids;
          PSID ResourceGroupDomainSid;
          ULONG ResourceGroupCount;
          [size_is(ResourceGroupCount)] PGROUP_MEMBERSHIP
     ResourceGroupIds;
     } KERB_VALIDATION_INFO;
```

Reserved fields are not defined in this document and are not used in the construction of access control tokens.

The fields are defined as follows:

KickOffTime - the time at which the server should forcibly logoff the client. If the client should not be forced off, this field should be set to (0x7fffffff,0xffffffff). If a kickoff time is to be enforced, the service ticket lifetime will never exceed this value.

UserId - This field contains the relative Id for the client. If zero, then the User ID is the first SID in the ExtraSids field.

PrimaryGroupId - This field contains the relative ID for this clientÆs primary group.

GroupCount - This field contains the number of groups, within the clientÆs domain, to which the client is a member.

GroupIds - This field contains an array of the relative Ids and attributes of the groups in the clientÆs domain of which the client is a member.

UserFlags - This field contains information about which fields in this structure are valid. The two bits that may be set are indicated below. Having these flags set indicates that the corresponding fields in the KERB_VALIDATION_INFO structure are present and valid.

```
#define LOGON_EXTRA_SIDS            0x0020
#define LOGON_RESOURCE_GROUPS       0x0200
```

LogonDomainId - This field contains the SID of the clientÆs domain. This field is used in conjunction with the UserId, PrimaryGroupId,and GroupIds fields to create the user and group SIDs for the client.

SidCount - This field contains the number of SIDs present in the ExtraSids field. This field is only valid if the LOGON_EXTRA_SIDS flag has been set in the UserFlags field.

ExtraSids - This field contains a list of SIDs for groups to which the user is a member. This field is only valid if the LOGON_EXTRA_SIDS flag has been set in the UserFlags field.
ResouceGroupCount - This field contains the number of resource groups in the ResourceGroupIds field. This field is only valid if the LOGON RESOURCE_GROUPS flag has been set in the UserFlags field._
ResourceGroupDomainSid - This field contains the SID of the resource domain. This field is used in conjunction with the ResourceGroupIds field to create the group SIDs for the client.
ResourceGroupIds - This field contains an array of the relative Ids and attributes of the groups in the resource domain of which the resource is a member.

When used in the KERB_VALIDATION_INFO, this is NDR encoded. The FILETIME type is defined as follows:

```
typedef unsigned int DWORD;

typedef struct _FILETIME {
    DWORD dwLowDateTime;
```



```
    DWORD dwHighDateTime;
} FILETIME;
```

Times are encoded as the number of 100 nanosecond increments since January 1, 1601, in UTC time.

When used in the KERB_VALIDATION_INFO, this is NDR encoded. The UNICODE_STRING structure is defined as:

```
typedef struct _UNICODE_STRING
    USHORT Length;
    USHORT MaximumLength;
    [size_is(MaximumLength / 2), length_is((Length) / 2) ]
USHORT *  Buffer;
} UNICODE_STRING;
```

The Length field contains the number of bytes in the string, not including the null terminator, and the MaximumLength field contains the total number of bytes in the buffer containing the string.

The GROUP_MEMBERSHIP structure contains the relative ID of a group and the corresponding attributes for the group.

```
typedef struct _GROUP_MEMBERSHIP {
    ULONG RelativeId;
```

```
        ULONG Attributes;
    } *PGROUP_MEMBERSHIP;
```

The group attributes must be:

```
    #define SE_GROUP_MANDATORY              (0x00000001L)
    #define SE_GROUP_ENABLED_BY_DEFAULT     (0x00000002L)
    #define SE_GROUP_ENABLED                (0x00000004L)
```

The SID structure is defined as follows:

```
    typedef struct _SID_IDENTIFIER_AUTHORITY {
        UCHAR Value[6];
    } SID_IDENTIFIER_AUTHORITY, *PSID_IDENTIFIER_AUTHORITY;
```

The constant value for the NT Authority is

```
    #define SECURITY_NT_AUTHORITY           {0,0,0,0,0,5}

    typedef struct _SID {
       UCHAR Revision;
       UCHAR SubAuthorityCount;
       SID_IDENTIFIER_AUTHORITY IdentifierAuthority;
       [size_is(SubAuthorityCount)] ULONG SubAuthority[*];
    } SID, *PSID;
```


Other authorities are defined in the Microsoft Developer Network Development Kit 3.
The SubAuthorityCount field contains the number of elements in the actual SubAuthority conformant array. The maximum number of subauthorities allowed is 15.

The KERB_SID_AND_ATTRIBUTES structure contains entire group SIDs and their corresponding attributes:

```
    typedef struct _KERB_SID_AND_ATTRIBUTES {
        PSID Sid;
        ULONG Attributes;
    } KERB_SID_AND_ATTRIBUTES, *PKERB_SID_AND_ATTRIBUTES;
```

The attributes are the same as the group attributes defined above.

5. Signatures (PAC_SERVER_CHECKSUM and PAC_PRIVSVR_CHECKSUM)

The PAC contains two digital signatures: one using the key of the

server, and one using the key of the KDC. The signatures are present for two reasons. First, the signature with the serverÆs key is present to prevent a client from generating their own PAC and sending it to the KDC as encrypted authorization data to be included in tickets. Second, the signature with the KDCÆs key is present to prevent an untrusted service from forging a ticket to itself with an invalid PAC. The two signatures are sent in PAC_INFO_BUFFERs of type PAC_SERVER_CHECKSUM and PAC_PRIVSVR_CHECKSUM respectively.

The signatures are contained in the following structure:

```
typedef struct _PAC_SIGNATURE_DATA {
    ULONG SignatureType;
    UCHAR Signature[1];
} PAC_SIGNATURE_DATA, *PPAC_SIGNATURE_DATA;
```

The fields are defined as follows:

SignatureType - This field contains the type of checksum used to create a signature. The checksum must be a keyed checksum.

Signature - This field consists of an array of bytes containing the checksum data. The length of bytes may be determined by the wrapping PAC_INFO_BUFFER structure.

For the serverÆs checksum, the key used to generate the signature should be the same key used to encrypt the ticket. Thus, if the enc_tkt_in_skey option is used, the session key from the serverÆs TGT should be used. The Key used to encrypt ticket granting tickets is used to generate the KDCÆs checksum.

The checksums are computed as follows:

1. The complete PAC is built, including space for both checksums
2. The data portion of both checksums is zeroed.
3. The entire PAC structure is checksummed with the serverÆs key, and the result is stored in the serverÆs checksum structure.
4. The serverÆs checksum is then checksummed with the KDC's key.
5. The checksum with the KDC key is stored in the KDC's checksum structure.

6. PAC Request Pre-Auth Data

Normally, the PAC is included in every pre-authenticated ticket received from an AS request. However, a client may also explicitly request either to include or to not include the PAC. This is done by

sending the PAC-REQUEST preauth data.

This is an ASN.1 encoded structure.

```
     KERB-PA-PAC-REQUEST     ::= SEQUENCE {
             include-pac[0] BOOLEAN -- if TRUE, and no pac present,
                                    -- include PAC.
                                    ---If FALSE, and pac
                                    -- PAC present, remove PAC
     }
```

The fields are defined as follows:

include-pac - This field indicates whether a PAC should be included or not. If the value is TRUE, a PAC will be included independent of other preauth data. If the value is FALSE, then no PAC will be included, even if other preauth data is present.

The preauth ID is:

```
     #define KRB5_PADATA_PAC_REQUEST         128
```

7. Security Considerations

Before the PAC data is used for access control, the PAC_SERVER_CHECKSUM signature MUST be checked. This will verify that the provider of the PAC data knows the server's secret key. Validation of the PAC_PRIVSVR_CHECKSUM is OPTIONAL. It is used to verify that the PAC was issued from the KDC and not placed in the ticket by someone other than the KDC with access to the service key.

Caution must be used with accepting the SIDs present in the logon-info part of the PAC. Only SIDs from a domain that is authoritative for a particular domain's SIDs should be used in the construction of access tokens. If a SID is found to be from outside of a domain's authoritative SID namespace, it MUST be ignored for purposes of access control.

8. References

1 Kohl, J., Neuman, C., "The Kerberos Network Authentication Service (V5)", RFC 1510, September 1993

2 Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997

3 Microsoft Developer's Network - http://msdn.microsoft.com

9. Author's Addresses

John Brezak
Microsoft Corporation
One Microsoft Way
Redmond, Washington
Email: jbrezak@microsoft.com

Full Copyright Statement