Secrets d'authentification épisode II
Kerberos contre-attaque

Aurélien Bordes
aurelien26@free.fr

Résumé L'authentification est un composant essentiel dans la sécurité des systèmes d'information.

Si de nombreux protocoles d'authentification coexistent, Kerberos s'est largement imposé ces dernières années comme le protocole d'authentification sur les réseaux locaux, en particulier avec son adoption comme service principal d'authentification dans les environnements Active Directory.

Cependant, Kerberos est un protocole complexe, tant au niveau de son fonctionnement (il faut jusqu'à 6 échanges pour réaliser une authentification) qu'au niveau de ses fonctionnalités (délégation, relation interdomaines, autorisation, etc.).

Cet article a deux objectifs. Le premier est d'étudier quelques spécificités du protocole Kerberos dans les environnements Active Directory et les impacts en matière de sécurité. Le second est de démontrer qu'en cas de compromission d'une base de comptes Active Directory (récupération par un attaquant de toutes les empreintes des mots de passe), les conséquences sont graves et que la remédiation est bien plus lourde qu'un simple changement des mots de passe de quelques utilisateurs.

1 Rappels sur Kerberos

1.1 Bases du protocole

Cette partie rappelle, via quelques explications simplifiées, des points notables et les fondements du protocole utiles pour la compréhension de la suite de l'article. La RFC 4120, la principale norme en vigueur sur Kerberos, constitue, dans tous les cas, la référence de toutes les descriptions données ci-dessous.

Tout d'abord, il existe *a minima* trois acteurs dans un scénario d'authentification via Kerberos :

- le client C qui souhaite s'authentifier auprès d'un serveur S ;
- le serveur S qui doit s'assurer de l'authenticité de C ;
- un tiers de confiance, le KDC.

Chaque acteur possède un secret, noté K_{ACTEUR}. Si chaque acteur connaît son propre secret (C connaît K_C, S connaît K_S et KDC connaît K_{KDC}), le tiers de confiance (c'est-à-dire le KDC) connaît également tous les secrets des entités de son royaume de confiance Kerberos (appelé *realm*). Ainsi, les acteurs partagent tous leur secret avec le KDC, qui connaît donc K_C et K_S. En réalité, chaque acteur ne connaît pas qu'une seule clé, mais un ensemble de clés de différents types leur permettant d'utiliser différents formats et algorithmes (MD5/DES, SHA-1/AES, etc.).

Note : il y a souvent, y compris dans la suite de ce document, confusion entre serveur et service. Dans la majorité des cas, un client souhaite se connecter sur une instance d'un service s'exécutant sur un serveur donné (exemple : service `cifs` sur `serveur-01.demo.test`). Dans ce cas, K_S fait référence à la clé du service `cifs` sur `serveur-01`. Cette confusion est amplifiée dans les environnements Active Directory où beaucoup de services (tels que le partage SMB de fichiers référencé par `cifs`) tournent sous les comptes de service *LocalSystem*[1] ou *NetworkService* qui reposent sur le compte de la machine dans le domaine. Ainsi, dans l'exemple de `cifs`, le service n'a pas de clé propre et K_S fait référence à la clé du compte de la machine sur laquelle le service s'exécute.

Les clés notées K sont des clés ayant une durée de vie importante (de quelques semaines à l'infini pour les clés n'expirant jamais). Il existe d'autres clés dites clés de sessions (notées $S_{ACTEURS}$) partagées entre deux acteurs (par exemple $S_{C,K}$ pour une clé partagée entre un client et le KDC). Ces clés sont négociées durant l'authentification d'un client et elles possèdent une durée de vie bien plus courte (quelques heures au maximum).

Kerberos définit plusieurs structures de données, dont deux sont particulièrement importantes :

- les **tickets**, générés par les KDC et transmis aux clients. Il existe deux types de tickets : les ***ticket-granting tickets*** (notés TGT dans la suite) et les **tickets de service**. Ceux-ci sont identiques dans leur forme, mais diffèrent par leur utilisation. Les clients conservent les tickets afin de les présenter par la suite aux KDC ou aux serveurs. Les tickets sont considérés comme publics, les parties sensibles étant systématiquement chiffrées ;
- les **authentifiants**, générés par les clients et transmis avec un ticket lors de l'authentification d'un client auprès d'un KDC ou d'un

1. Affiché également comme *SYSTEM*.

serveur. C'est la capacité à chiffrer correctement un authentifiant avec une clé de session qui prouve l'authenticité d'un client.

En simplifiant, un ticket est ainsi composé :

```
Ticket =
  sname et realm : nom du serveur
  enc-part (partie chiffrée) =
    flags : options du ticket
    key : clé contenue dans le ticket
    cname et crealm : nom du client
    authtime, starttime et endtime : période de validité
    caddr : adresse du client
    authorization-data : données d'autorisation
```

Un ticket fait donc référence à deux clés : la première est celle qui chiffre la partie chiffrée du ticket (enc-part) et la seconde est celle contenue dans la partie chiffrée du ticket (champ `enc-part.key`).

Quant aux authentifiants, toujours en simplifiant, leur description est donnée ci-dessous. Il est important de noter que les authentifiants sont systématiquement intégralement chiffrés. Outre l'authentification, les authentifiants assurent la protection contre le rejeu des messages.

```
Authenticator =
  cname et crealm : nom du client
  ctime : temps du client (pour l'anti-rejeu)
  seq-number : numéro de séquence
```

1.2 Échanges Kerberos

L'authentification d'un client auprès d'un serveur va s'opérer en 3 échanges auprès de 3 services distincts :

- ***Authentication Service*** (AS) qui permet au client :
  - de récupérer un ticket particulier, le TGT,
  - de négocier une clé de session avec le KDC ($\mathtt{S_{C,K}}$) ;
- ***Ticket-Granting Service*** (TGS) qui permet au client :
  - de s'authentifier auprès du KDC en lui transmettant un TGT et un authentifiant chiffré avec la clé de session ($\mathtt{S_{C,K}}$) qu'il partage avec le KDC,
  - de récupérer un ticket de service pour un service donné,
  - de négocier une clé de session ($\mathtt{S_{C,S}}$) avec le service correspondant à celui demandé pour le ticket de service ;

– **Client/Server Authentication** (messages associés notés AP) qui permet au client :
  – de s'authentifier auprès d'un serveur en lui transmettant un ticket de service TS et un authentifiant chiffré avec la clé de session ($S_{C,S}$) qu'il partage avec le service,
  – optionnellement d'authentifier le serveur dans le cadre de l'authentification mutuelle.

Chaque étape est composée d'une requête `REQ` et d'une réponse `REP`, ce qui fait 6 échanges pour réaliser une authentification complète :

– `KRB_AS_REQ` et `KRB_AS_REP` ;
– `KRB_TGS_REQ` et `KRB_TGS_REP` ;
– `KRB_AP_REQ` et `KRB_AP_REP`.

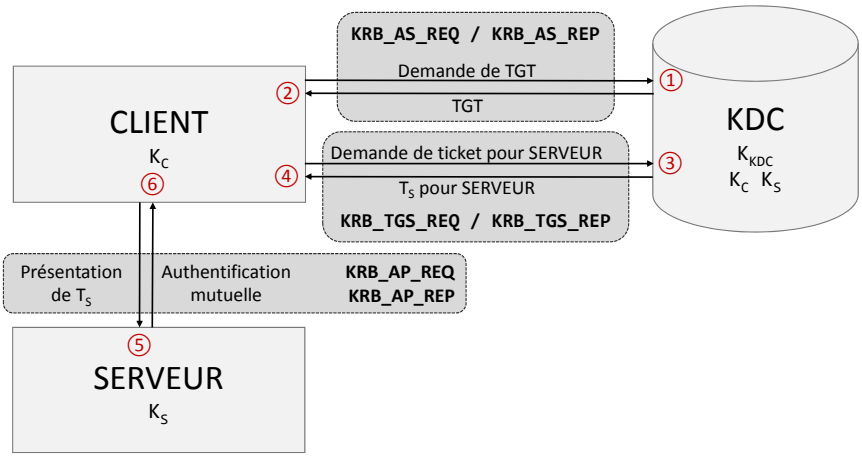

Figure 1. Synopsis d'une authentification Kerberos

Les chapitres suivants illustrent ces 6 échanges dans le cas de l'authentification d'un utilisateur (`Administrateur`) auprès d'un service (`cifs/serveur-01.demo.test`) en environnement Active Directory. Les noms des structures et des champs sont ceux définis par la RFC. Dans tous les exemples de message, les champs préfixés par (`*`) sont marqués comme optionnels et ceux préfixés par ($\|$) sont dans des parties chiffrées. L'intérêt de ces exemples, par rapport à une capture réseau classique, est de présenter les parties chiffrées.

1.3 Message KRB_AS_REQ

Un message KRB_AS_REQ (listing 1) permet à un client de demander un TGT à un serveur d'authentification (*Authentication Service*) d'un KDC. Il s'agit d'un message générique KRB_KDC_REQ de demande de ticket.

```
[AS-REQ]
  pvno : numéro de version, toujours 5
  msg-type : type de message, 10 pour KRB_AS_REQ
  *padata: Séquence de blocs PA-DATA (voir ci-dessous)
    [PA-ENC-TIMESTAMP] : EncryptedData[PA-ENC-TS-ENC]
      [PA-ENC-TS-ENC] == Partie chiffrée par Kc ==
      patimestamp : 20140327121101
  req-body:
    [KDC-REQ-BODY]
      kdc-options: 0x40810010 - Options du ticket demandées par le client
        (enc_pa_rep, renewable, forwardable)
      *cname: Administrateur (1) - Nom du client
      realm: DEMOTEST
      *sname: krbtgt/DEMO.TEST (2) - Nom du service demandé
        (toujours krbtgt/REALM pour les TGT)
      till: 2037/09/13 02:48:05.0
      *rtime: 2037/09/13 02:48:05.0
      nonce: 0x5ec10613
      etype: Séquences des identifiants des algorithmes de chiffrement
             supportés par le client
        →0x18 (aes256-cts-hmac-sha1-96)
        →0x17 (aes128-cts-hmac-sha1-96)
        →0x23 (rc4-hmac)
        →0x24 (rc4-hmac-exp)
        →0xffffff79 (rc4-hmac-old-exp)
        →0x3 (des-cbc-md5)
      *addresses: Séquence des adresses du client
        →addr-type: 20 (NetBios), address: WIN8
```

Listing 1. Message KRB_AS_REQ

Il est à noter que :

- le champ padata contient, optionnellement, des blocs de type PreAuthentication (PA*) qui permettent d'étendre les messages de demande ou de réponse de ticket (KRB_KDC_REQ et KRB_KDC_REP) ;
- le nom du client est le nom de l'utilisateur Windows. Dans l'annuaire Active Directory, il doit correspondre à l'attribut userPrincipalName d'un compte ou, à défaut, à l'attribut samAccountName ;
- dans le cas d'une requête sur le service AS, le nom du service (sous la forme d'un SPN [2]) demandé est toujours krbtgt/REALM où REALM correspond au nom du domaine Active Directory ;

2. SPN : *service principal name*. Nom unique d'identification d'un service dans un domaine Kerberos. Généralement de la forme service/serveur@realm.

– sous Windows, l'adresse émise par le client correspond au nom NetBios de sa machine.

Les identifiants des algorithmes de chiffrement du champ `etype` sont décrits dans la partie 2.1.

De base, un message `KRB_AS_REQ` permet de solliciter l'obtention d'un TGT et d'une clé de session pour n'importe quel utilisateur sans aucune forme d'authentification préalable. Or, dans la réponse, la clé de session est chiffrée par celle de l'utilisateur. Ceci peut permettre à un attaquant qui aurait écouté le trafic réseau de tenter des attaques hors ligne sur le mot de passe de l'utilisateur.

Afin d'éviter ce scénario, un KDC peut exiger du client qu'il prouve sa connaissance de la clé K_C. Pour cela, le client doit inclure dans sa requête un bloc `PA-DATA` de type `PA-ENC-TIMESTAMP`. Ce bloc contient l'heure du client chiffrée par sa clé K_C. Le KDC valide ce bloc en vérifiant si la clé correspond bien à celle du client. Par ailleurs, la différence entre l'heure chiffrée par le client (champ `patimestamp`) et celle du KDC ne doit pas excéder 5 minutes (cf. également partie 1.9).

Évidemment, cette forme d'authentification n'est pas suffisante et est vulnérable à un rejeu dans la période de tolérance des 5 minutes. On parle alors de préauthentification, permettant simplement de ne pas permettre à n'importe qui de demander un TGT arbitrairement, sans aucune connaissance préalable.

Au niveau de l'Active Directory, par défaut, tous les comptes doivent se préauthentifier. Ceci peut être désactivé pour un compte donné en positionnant l'option `ADS_UF_DONT_REQUIRE_PREAUTH` sur l'attribut `userAccountControl` [23] (ce qui correspond à l'option « La pré-authentification Kerberos n'est pas nécessaire » dans l'interface graphique).

1.4 Message KRB_AS_REP

Pour répondre au client, un message de type `KRB_AS_REP` (listing 2) est envoyé. Comme pour le cas précédent, il s'agit d'un message générique de type `KRB_KDC_REP` qui permet au client de récupérer son TGT (dont la partie chiffrée, par K_{KDC}, contient $S_{C,K}$) ainsi que $S_{C,K}$ dans la partie chiffrée par K_C.

```
[AS-REP]
 pvno : numéro de version, toujours 5
 msg-type : type de message, 11 pour KRB_AS_REP
 *padata: Séquence de blocs PA-DATA
   [PA-ETYPE-INFO2]
    etype: 0x18 (aes256-cts-hmac-sha1-96)
    salt: DEMO.TESTAdministrateur
 crealm: DEMO.TEST
 cname: Administrateur (1)
 ticket:
   [Ticket] TGT transmis au client
    tkt-vno: numéro de version, toujours 5
    realm: DEMO.TEST
    sname: krbtgt/DEMO.TEST (2)
    enc-part: EncryptedData[EncTicketPart] etype: 18, kvno: 2
      [EncTicketPart] == Partie chiffrée par K_KDC ==
       flags: 0x40e10000
        (enc_pa_rep, pre_authent, initial, renewable, forwardable)
       key: [EncryptionKey] keytype: 0x12, keyvalue:
         Clé de session S_C,K contenue dans le ticket
       crealm: DEMO.TEST
       cname: Administrateur (1)
       authtime: 2014/03/27 11:11:58.0
       *starttime: 2014/03/27 11:11:58.0
       endtime: 2014/03/27 21:11:58.0
       *renew-till: 2014/04/03 11:11:58.0
       *authorization-data: cf. partie 2.5
 enc-part: EncryptedData[EncASRepPart] etype: 23, kvno: 1
   [EncKDCRepPart] == Partie chiffrée par K_C ==
    key: [EncryptionKey] keytype: 0x12, keyvalue:
      Clé de session S_C,K transmise au client
    nonce 0x5ec10613 - aléa repris de la requête du client
    *key-expiration: 2037/09/14 02:48:05.0
    flags: 0x40e10000
      (enc_pa_rep, pre_authent, initial, renewable, forwardable)
    authtime: 2014/03/27 11:11:58.0
    starttime: 2014/03/27 11:11:58.0
    endtime: 2014/03/27 21:11:58.0
    *renew-till: 2014/04/03 11:11:58.0
    srealm: DEMO.TEST
    sname: krbtgt/DEMO.TEST (2)
    *caddr:
      →0x18 addr-type: 20 (NetBios), address: WIN8
```

Listing 2. Message KRB_AS_REP

Si le compte qui sollicite un ticket supporte le chiffrement au moyen de clés nécessitant une graine (*salt*), celle-ci est envoyée au client via une séquence d'un ou plusieurs blocs PA-PW-SALT (pour les clés de type DES) ou PA-ETYPE-INFO2 (pour les clés de type AES, ce bloc permettant également d'envoyer le nombre d'itérations). Ces mêmes blocs sont également envoyés au client en cas d'erreur (absence ou erreur de

préauthentification, utilisation d'algorithme de chiffrement non géré, etc.) dans des messages de type `KRB_ERROR` (cf. partie 2.4).

Hormis le nom du serveur associé au ticket (`sname` et `realm`), tous les autres champs du ticket sont inaccessibles au client car chiffrés par K_{KDC}. Afin de permettre au client d'obtenir de manière sécurisée les informations nécessaires, la partie chiffrée de la réponse (`enc-part` chiffrée par la clé K_C, et donc accessible au client) duplique de nombreux champs du ticket, en particulier :

- la clé de session $S_{C,K}$ partagée avec le KDC ;
- les champs `flags`, `authtime`, `starttime`, `endtime`, `renew-till` et `caddr`.

Certains de ces champs sont par ailleurs repris du message `KRB_AS_REQ` afin de s'assurer qu'ils n'ont pas été modifiés (`nonce`, `caddr`, etc.).

Après ces deux échanges, le client dispose d'un TGT et d'une clé de session $S_{C,K}$ qu'il conserve dans les données de sa session d'authentification (cf. partie 2.4). Quant au KDC, il ne conserve aucune information : la clé de session $S_{C,K}$ sera retrouvée dans le TGT que devra lui présenter le client par la suite.

1.5 Message KRB_TGS_REQ

Un message `KRB_TGS_REQ` (listing 3) est identique, dans sa forme, à un message `KRB_AS_REQ` vu précédemment. Il s'agit d'un message de demande de ticket de type `KRB_KDC_REQ` avec cependant quelques différences notables :

- ce n'est pas un TGT qui est demandé (avec le SPN spécifique `krbtgt/REALM`), mais un ticket de service dont le client spécifie le nom de l'instance sous la forme service/serveur (par exemple : `cifs/serveur.domaine.demo`) ;
- le ticket est demandé au service TGS du KDC qui impose l'authentification du demandeur contrairement au service AS. Pour cela, le bloc `PA-ENC-TIMESTAMP` (si présent) est remplacé par un bloc `PA-TGS-REQ` (obligatoirement présent). Ce bloc contient un message `KRB_AP_REQ` dont le fonctionnement est décrit dans la partie 1.7 et qui permet :
  - de transmettre le TGT du client au KDC,
  - d'authentifier auprès du KDC le client en prouvant que celui-ci connaît la clé de session $S_{C,K}$ associée au TGT,
  - d'assurer l'anti-rejeu de la demande.

```
[TGS-REQ]
| pvno : numéro de version, toujours 5
| msg-type : type de message, 12 pour KRB_TGS_REQ
| *padata: Séquence de blocs PA-DATA
|   [AP-REQ] - Message KRB_AP_REQ authentifiant le client auprès du service TGS
|   | pvno: 5
|   | msg-type: 14
|   | ap-option: 0x0 (pas d'option)
|   | ticket:
|   |   [Ticket] TGT transmis du client au KDC (extraits, voir AS-REP)
|   |   | ...
|   |   | sname: krbtgt/DEMO.TEST (2)
|   |   | enc-part: EncryptedData[EncTicketPart] etype: 18, kvno: 2
|   |   |   [EncTicketPart] == Partie chiffrée par K_KDC ==
|   |   |   || ...
|   |   |   || key: [EncryptionKey] keytype: 0x12, keyvalue:
|   |   |   ||   Clé de session S_C,K contenue dans le ticket
|   |   |   || ...
|   | authenticator: EncryptedData[Authenticator] etype: 18
|   |   [Authenticator] == Partie chiffrée par S_C,K ==
|   |   || authenticator-vno: 5
|   |   || crealm: DEMO.TEST
|   |   || cname: Administrateur (1)
|   |   || *cksum:
|   |   ||   [Checksum]
|   |   ||   | cksumtype: 0x7
|   |   ||   | checksum: f166b445ae30928f9e74874d60c94e83
|   |   || ctime: 2014/03/27 11:12:00.0
|   |   || *seq-number: 61eb3541
| req-body:
|   [KDC-REQ-BODY]
|   | kdc-options: 0x40810000 - Options de ticket demandée par le client
|   |   (enc_pa_rep, renewable, forwardable)
|   | realm: DEMO.TEST
|   | *sname: cifs/dc-2012-01.demo.test (2) - Nom du service demandé
|   | till: 2037/09/13 02:48:05.0
|   |   Date de validité maximale proposée par le client pour le ticket
|   | nonce: 0x61eb3541
|   | etype: Séquences d'identifiant des algorithmes de chiffrement
|   |        supportés par le client
|   |   →0x18 0x18 (aes256-cts-hmac-sha1-96)
|   |   →0x18 0x17 (aes128-cts-hmac-sha1-96)
|   |   →0x18 0x23 (rc4-hmac)
|   |   →0x18 0x24 (rc4-hmac-exp)
|   |   →0x18 0xffffff79 (rc4-hmac-old-exp)
|   | *enc-authorization-data: cf. partie 2.5
```

Listing 3. Message KRB_TGS_REQ

1.6 Message KRB_TGS_REP

Là encore, un message KRB_TGS_REP (listing 4) est identique, dans sa forme, à un message KRB_AS_REP (message de type KRB_KDC_REP) et vise

à transmettre au client un ticket de service ainsi que la clé de session $S_{C,S}$ associée.

```
[TGS-REP]
  pvno : numéro de version, toujours 5
  msg-type : type de message, 13 pour KRB_TGS_REP
  crealm: DEMO.TEST
  cname: Administrateur (1)
  ticket:
    [Ticket] Ticket de service transmis au client
      tkt-vno: 5
      realm: DEMO.TEST
      sname: cifs/dc-2012-01.demo.test (2) - SPN du service
      enc-part: EncryptedData[EncTicketPart] etype: 18, kvno: 10
        [EncTicketPart] == Partie chiffrée par K_S ==
          flags: 0x40a10000
            (enc_pa_rep, cname_in_pa_data, pre_authent, renewable, forwardable)
          key: [EncryptionKey] keytype: 0x12, keyvalue:
            Clé de session S_C,S contenue dans le ticket
          crealm: DEMO.TEST
          cname: Administrateur (1)
          authtime: 2014/03/27 11:11:58.0 - Date de validité du ticket
          *starttime: 2014/03/27 11:12:00.0
          endtime: 2014/03/27 21:11:58.0
          *renew-till: 2014/04/03 11:11:58.0
          *authorization-data: Données d'autorisation, cf. partie 2.5
  enc-part: EncryptedData[EncTGSRepPart] etype: 18
    [EncKDCRepPart] == Partie chiffrée par S_C,K ==
      key: [EncryptionKey] keytype: 0x12, keyvalue:
        Clé de session S_C,S transmise au client
      nonce 0x61eb3541 - Aléa repris de la demande du client
      flags: 0x40a10000
        (enc_pa_rep, cname_in_pa_data, pre_authent, renewable, forwardable)
      authtime: 2014/03/27 11:11:58.0
      starttime: 2014/03/27 11:12:00.0
      endtime: 2014/03/27 21:11:58.0
      *renew-till: 2014/04/03 11:11:58.0
      srealm: DEMO.TEST
      sname: cifs/dc-2012-01.demo.test (2)
```

Listing 4. Message KRB_TGS_REP

1.7 Message KRB_AP_REQ

Un message KRB_AP_REQ (listing 5) permet à un client de s'authentifier auprès du TGS d'un KDC (comme ci-dessus) ou de tout autre service acceptant Kerberos comme protocole d'authentification. Préalablement à tout message KRB_AP_REQ, le client doit être en possession d'un ticket (TGT ou de service) et de la clé de session associée. C'est en prouvant au serveur qu'il connaît cette clé de session que le client prouve son identité.

Un message KRB_AP_REQ assure trois finalités :

– transférer le ticket du client au service. Celui-ci étant chiffré par la clé du service destinataire (K_{KDC} pour le TGS ou K_S ou pour un service), le serveur a accès aux informations chiffrées du ticket. Il est alors en mesure de récupérer la clé de session contenue dans le ticket ($S_{C,K}$ ou $S_{C,S}$) ;
– permettre au client de s'authentifier auprès du service. Pour cela, le client doit prouver qu'il connaît la clé de session contenue dans le ticket. Cette preuve est apportée en chiffrant l'authentifiant contenu dans le champ `authenticator` de la requête ;
– se protéger contre le rejeu.

```
[AP-REQ]
  pvno : numéro de version, toujours 5
  msg-type : type de message, 14 pour KRB_AP_REQ
  ap-option: 0x20000000 - Options du message AP-REQ demandées par le client
    (mutual-required)
  ticket:
    [Ticket] Ticket de service transmis du client au serveur
      ...
      sname: cifs/dc-2012-01.demo.test (2)
      enc-part: EncryptedData[EncTicketPart] etype: 18, kvno: 10
        [EncTicketPart] == Partie chiffrée par K_S ==
          ...
          key: [EncryptionKey] keytype: 0x12, keyvalue:
            Clé de session S_C,S contenue dans le ticket
  authenticator: EncryptedData[Authenticator] etype: 18
    [Authenticator] == Partie chiffrée par S_C,S ==
      authenticator-vno: 5
      crealm: DEMO.TEST
      cname: Administrateur (1) - Nom du client
      *cksum:
        [Checksum]
          cksumtype: 0x7
          checksum: 42ab124f520aee347b379039c490934
      ctime: 2014/03/27 11:12:00.0 - Temps actuel du client
      *subKey: [EncryptionKey] keytype: 0x12, keyvalue:
        Clé de session C->S supplémentaire
      *seq-number: 61ea6f95 - Numéro de séquence
      *authorization-data: Données d'autorisation, cf. partie 2.5
```

Listing 5. Message `KRB_AP_REQ`

Le client, en spécifiant l'option `mutual-required` dans le champ `ap-options`, peut demander l'authentification mutuelle. Dans ce cas, le serveur doit également s'authentifier auprès du client en prouvant qu'il connaît K_S. Ceci est réalisé par l'envoi, du serveur au client, d'un message `KRB_AP_REP`.

L'authentification mutuelle est un apport non négligeable par rapport à de nombreux protocoles d'authentification, en particulier NTLM. Cependant, la demande d'authentification mutuelle doit être explicitement formulée par le client (via l'option `ISC_REQ_MUTUAL_AUTH` de la fonction `InitializeSecurityContext`) et vérifiée.

1.8 Message KRB_AP_REP

Un message de type `KRB_AP_REP` (listing 6) permet à un serveur de s'authentifier auprès d'un client en prouvant qu'il connaît K_S. Cette preuve se réalise par deux opérations :

1. le serveur récupère, dans la partie chiffrée par K_S du ticket de service présenté par le client, la clé $S_{C,S}$;
2. le serveur chiffre le message `KRB_AP_REP` avec la clé de session $S_{C,S}$.

```
[AP-REP]
  pvno: numéro de version, toujours 5
  msg-type: type de message, 15 pour KRB_AP_REP
  enc-part: EncryptedData[EncAPRepPart] etype: 18
    [EncAPRepPart] == Partie chiffrée par S_C,S ==
      ctime: 2014/03/27 11:12:00.0
      *subkey: [EncryptionKey] keytype: 0x12, keyvalue:
        Clé de session S<->C supplémentaire
      *seq-number: 0x61c8098c
```

Listing 6. Message `KRB_AP_REP`

Si le client déchiffre correctement la partie chiffrée (`enc-part`) et que l'heure ne s'écarte pas de plus de 5 minutes par rapport à sa propre heure, il est assuré que le serveur connaît $S_{C,S}$ et donc K_S. Il est également possible de négocier une nouvelle clé de session qui servira dans les extensions de chiffrement et d'authentification (fonctions `MakeSignature`, `EncryptMessage`, etc.).

1.9 Protections contre le rejeu

Il est important que les messages Kerberos soient protégés contre le rejeu et tout particulièrement les messages `KRB_AP_REP`. Si tel n'était pas le cas, il suffirait à un attaquant de capturer sur le réseau une authentification présentée par un client auprès d'un service puis de la rejouer sur ce même service.

Pour se prémunir contre ce type d'attaques, plusieurs mécanismes sont mis en œuvre.

Le premier mécanisme, macroscopique, consiste à horodater toutes les demandes des clients. Ainsi, le client doit chiffrer son heure actuelle (date, heure, minute, seconde et microseconde) et l'envoyer au serveur (champ `patimestamp` du bloc `PA-ENC-TIMESTAMP` ou champ `ctime` de l'authentifiant). Le serveur accepte les demandes uniquement si l'heure du client et la sienne ne différent pas de plus de 5 minutes[3]. Si la différence est supérieure, une dérive de temps (*clock skew*) est détectée et les requêtes sont rejetées (`KRB_AP_ERR_SKEW`). Ce mécanisme impose la synchronisation des heures des systèmes de tous les acteurs prenant part à l'authentification Kerberos.

Le deuxième mécanisme, microscopique, est la mise en place d'un cache des authentifiants reçus. Ainsi, sous Windows, l'empreinte MD5 de tous les authentifiants reçus par un message `KRB_AP_REP` est calculée et mise dans ce cache. L'unicité des empreintes est assurée par les numéros de séquence des authentifiants (champ `seq-number`) qui doivent toujours être différents. Avant de traiter un authentifiant, le système regarde si son empreinte n'est pas déjà présente dans ce cas. Si tel est le cas, le message `KRB_AP_REP` est rejeté (`KRB_AP_ERR_REPEAT`).

Le troisième mécanisme concerne les tickets (TGT ou tickets de service). Afin d'éviter qu'un attaquant, ayant dérobé (cf. partie 2.4) un ticket et la clé associée, puisse les réutiliser depuis une autre machine, le KDC peut spécifier l'adresse du client dans la partie chiffrée du ticket (champ `caddr`). Ainsi, si un attaquant tente d'utiliser un tel ticket depuis une autre machine (dont l'adresse est supposée être différente de celle contenue dans le ticket), le ticket et donc la demande sont rejetés.

Cependant, lier les tickets Kerberos à l'adresse IP des machines n'est pas sans poser de difficultés, en particulier dans les environnements où la traduction d'adresse (NAT) ou l'adressage dynamique (DHCP) sont mis en œuvre. En effet, le changement de l'adresse IP de la machine d'un client invaliderait tous ses tickets. Ainsi, dans une configuration par défaut, les contrôleurs de domaine n'incluent pas l'adresse IP du client

3. Cette durée, qui peut être changée, est celle par défaut sous Windows conformément à la recommandation de la RFC.

dans les tickets qu'ils émettent.

Il est possible de modifier ces comportements par défaut, via la base de registre [11], pour demander :

- aux clients d'inclure leur adresse IP dans leur requête de ticket (`ClientIpAddress`) ;
- aux KDC d'inclure l'adresse IP aux TGT (`KdcUseClientAddresses`) et de vérifier les adresses dans les demandes de tickets qui leur sont adressées (`KdcDontCheckAddresses`).

2 Adaptation à l'environnement Active Directory

2.1 Algorithmes

Comme vu précédemment, chaque acteur dispose d'un ensemble de clés partagées avec le KDC. Toutes ces clés sont dérivées du mot de passe de l'acteur via des fonctions de type *string-to-key*. Les mots de passe peuvent être choisis, ce qui est généralement le cas pour les comptes utilisateur (par exemple $\mathtt{K_C}$), ou générés automatiquement dans le cas des comptes de service ou des comptes machine (par exemple $\mathtt{K_{KDC}}$ et $\mathtt{K_S}$).

Une clé est définie par son type, sa valeur et son numéro de version. Le type indique l'algorithme de dérivation du mot de passe. Initialement, la première norme de Kerberos (RFC 1510) spécifiait qu'*a minima* des clés de chiffrement de type DES devaient être supportées par chaque acteur (pour les algorithmes `des-cbc-crc`, `des-cbc-md4` et `des-cbc-md5`).

Lors de la conception de l'Active Directory à la fin des années 90, Microsoft ne pouvait utiliser uniquement DES pour le chiffrement. En effet, dans les bases de comptes SAM des domaines Windows NT, les empreintes des mots de passe, au format NTLM qui repose sur MD4, étaient calculées différemment et ne pouvaient servir de clé de type DES.

Microsoft a donc choisi, dans Windows 2000, d'introduire d'autres algorithmes basés sur RC4 pour le chiffrement et utilisant les empreintes NTLM (RFC 4757). Ceci permettait de disposer, lors d'une migration d'une base SAM d'un domaine Windows NT en annuaire Active Directory, de clés Kerberos en réutilisant les empreintes NTLM existantes. Par la suite, lorsque les mots de passe étaient changés, les clés DES étaient calculées afin de permettre l'utilisation des algorithmes utilisant ce type de clé.

Plus récemment, de nouveaux algorithmes jugés plus sûrs ont été adoptés tels que la dérivation de mots de passe basée sur SHA-1 et PBKDF2 (avec itérations multiples) ou l'algorithme de chiffrement AES (RFC 3962). Ces nouveaux standards sont supportés depuis Windows Vista et Windows Server 2008.

Le tableau 1 recense les principaux algorithmes mis en œuvre dans les environnements Active Directory. Si les précédentes normes de Kerberos laissaient une certaine imprécision, la RFC 4120 précise bien que l'identifiant d'un type de clé spécifie également l'algorithme de chiffrement associé. On peut remarquer que beaucoup d'algorithmes reposent encore sur l'empreinte NTLM bien que cette forme d'empreinte ne mette pas en œuvre l'utilisation de graine ou d'itérations dans sa dérivation depuis un mot de passe.

Table 1. Liste des types de clés et des algorithmes de chiffrement

Type	Nom	Itérations	Graine	Norme
0x1	des-cbc-crc	N/A	OUI	RFC 3961
0x3	des-cbc-md5	N/A	OUI	RFC 3961
0x11	aes128-cts-hmac-sha1-96	4096	OUI	RFC 3962
0x12	aes256-cts-hmac-sha1-96	4096	OUI	RFC 3962
0x17	rc4-hmac	N/A	N/A	RFC 4757 (empreinte NTLM)
0x18	rc4-hmac-export	N/A	N/A	RFC 4757 (empreinte NTLM)
0xffffff79	rc4-hmac-old-export	N/A	N/A	RFC 4757 (empreinte NTLM)
0xffffff7b	rc4-hmac-old	N/A	N/A	RFC 4757 (empreinte NTLM)
0xffffff80	rc4-md4	N/A	N/A	RFC 4757 (empreinte NTLM)

Le nombre d'itérations indique celui par défaut. Quant à la graine, elle est toujours, par défaut, la concaténation du nom du domaine et du nom du compte.

2.2 Clés des KDC

Pour stocker les clés $\mathtt{K}_{\mathtt{KDC}}$, l'Active Directory utilise un compte d'un utilisateur dénommé `krbtgt`. Il s'agit d'un simple compte, désactivé, sans droit (au sens permission dans l'Active Directory) ni privilège système particulier.

Si plusieurs contrôleurs de domaine sont utilisés, la réplication de l'annuaire assure que chaque contrôleur dispose de clés $\mathtt{K}_{\mathtt{KDC}}$ partagées

(c'est-à-dire identiques). Ceci est indispensable pour s'assurer que n'importe quel contrôleur de domaine puisse jouer le rôle de KDC et qu'un TGT demandé sur un contrôleur puisse être présenté et déchiffré par un autre contrôleur.

Enfin, lorsque des contrôleurs de domaine en lecture seule (RODC pour *Read-Only Domain Controller*) sont mis en œuvre, chacun dispose d'une clé K_{KDC} propre et associée au compte `krbtgt_<RodcID>`.

2.3 Stockage des clés dans l'AD

Pour chaque compte (utilisateur, machine ou `krbtgt`), plusieurs attributs de l'annuaire Active Directory sont employés pour stocker les empreintes des mots de passe ou des clés Kerberos.

Les empreintes LM [4] et NTLM sont stockées respectivement dans les attributs `dBCSPwd` et `unicodePwd` et leur historique réciproque dans les attributs `lmPwdHistory` et `ntPwdHistory`. Comme vu dans la partie 2.1, plusieurs types de clés Kerberos et d'algorithmes de chiffrement reposent sur l'empreinte NTLM (rc4-hmac, rc4-hmac-old, etc.).

Pour le stockage des clés aux formats historiques basées sur DES, le champ `supplementalCredentials` est utilisé [20]. Ce champ est une structure binaire pouvant contenir diverses formes de stockage du mot de passe (baptisés *packages*) allant du mot de passe en clair (si activé) aux formes d'empreintes utilisées par Kerberos.

Le package `Primary:Kerberos` contient ainsi une structure `KERB_STORED_CREDENTIAL` qui contient, à son tour, les formes DES des clés (champ `Credentials`) et d'autres informations telles que :

- la graine utilisée pour le calcul des clés ;
- les clés de type `des-cbc-md5` et `des-cbc-crc` calculées à partir du mot de passe courant de l'utilisateur ;
- les mêmes types de clés, mais calculées avec le précédent mot de passe (champ `OldCredentials`).

4. Depuis Windows Server 2008, dans une configuration par défaut, les empreintes LM ne sont plus générées. Les champs `dBCSPwd` et `lmPwdHistory` sont donc vides et les algorithmes basés sur ce format d'empreinte ne peuvent plus être utilisés.

Avec Windows 2008, et le support des nouveaux algorithmes, un nouveau package est apparu : `Primary:Kerberos-Newer-Keys`. Celui-ci contient une structure `KERB_STORED_CREDENTIAL_NEW` qui permet, par rapport à la structure précédente :

- de stocker les formes des clés `aes128-cts-hmac-sha1-96` et `aes256-cts-hmac-sha1-96` ;
- de stocker le nombre d'itérations utilisées dans le calcul des clés, ce qui permet de faire varier ce nombre qui, par défaut, est fixé à 4096 ;
- de stocker les clés correspondant au mot de passe courant de l'utilisateur (champ `Credentials`), ainsi que celles calculées à partir du mot de passe précédent (champ `OldCredentials`) et de celui d'avant (champ `OlderCredentials`).

Enfin, à chaque clé Kerberos est associé un numéro de version qui permet, en cas de changement de mot de passe, de pouvoir déterminer la clé ayant été utilisée pour chiffrer les données. Au niveau des échanges Kerberos, chaque partie chiffrée (structure de type `EncryptedData`) spécifie donc le type d'algorithme (champ `etype`) et le numéro de clé (champ `kvno`). L'Active Directory utilise quant à lui l'attribut `msDS-KeyVersionNumber` pour y stocker cette information [5]. Celui-ci contient un compteur qui est incrémenté à chaque changement de mot de passe du compte associé.

Si le mode de fonctionnement avec l'utilisation des numéros de version est celui de l'esprit de la RFC, il en est tout autrement dans les systèmes Windows et l'Active Directory. En effet, comme décrit dans [29], dans ces environnements, le champ `kvno` est tout simplement ignoré [6] par les clients et les KDC. Ceux-ci conservant systématiquement les clés correspondant aux mots de passe courant et précédent, le processus de sélection des clés de déchiffrement est différent. Une tentative de déchiffrement du bloc chiffré à l'aide des clés associées au mot de passe courant est d'abord réalisée. Si cela échoue, ce sont les clés correspondant au mot de passe précédent qui sont essayées. C'est seulement si les deux tentatives échouent que le mot de passe est considéré comme invalide.

Il est important de rappeler que tous les attributs utilisés pour stocker des empreintes ou des clés (`unicodePwd`, `supplementalCredentials`)

5. Lorsque des RODC sont mis en œuvre, ce numéro sert également à désigner le RODC [28].

6. Il est tout de même renseigné par les KDC dans leurs réponses.

sont de type `SecretAttribute` [8] et ne peuvent, à ce titre, jamais être extraits, par exemple par des requêtes LDAP.

Au niveau des comptes de l'Active Directory :
- les comptes utilisateur contiennent les clés des utilisateurs ;
- les comptes machine contiennent les clés des serveurs (ce qui correspond également aux services qui tournent sous l'identité *LocalSystem* et *NetworkService*) ;
- le compte `krbtgt` contient la clé K_{KDC}.

Par exemple, pour un compte baptisé « SSTIC », un extrait des informations conservées par l'Active Directory est donné dans le listing 7. Il faut rappeler que, pour des raisons de compatibilité, l'empreinte NTLM n'a pas de graine contrairement à toutes les autres clés Kerberos (`des-cbc-md5`, `aes256-cts-hmac-sha1-96`, etc.) qui, elles, sont calculées avec une graine qui dépend, par défaut, du nom du domaine et du compte.

```
cn: SSTIC
objectClass: top; person; organizationalPerson; user
userAccountControl: 0x200 (NORMAL_ACCOUNT)
userPrincipalName: sstic@demo.test
dBCSPwd: <VIDE>
unicodePwd: 78D53C7B319AE96D741A99D79477B8A3
            (empreinte NTLM du mot de passe courant)
lmPwdHistory : <VIDE>
ntPwdHistory: (historique des empreintes NTLM)
  78D53C7B319AE96D741A99D79477B8A3, F60978608DE49640DB689DF310A7DC7A, ↩
  BFFAB41FB7E427F264B9304FE4ADAED3, D05DA968FF0159374C2258D6060CEA96
supplementalCredentials:
  USER_PROPERTIES.Length: 0x9d8
                 .PropertySignature: 0x50, 0x00 (constante)
                 .PropertyCount: 4
                 .UserProperties:
    [0] USER_PROPERTY.PropertyName: "Primary:Kerberos-Newer-Keys"
        KERB_STORED_CREDENTIAL_NEW.Revision: 4
                                  .DefaultIterationCount: 0x1000 (4096)
                                  .DefaultSalt: "DEMO.TESTsstic"
                                  .CredentialCount: 3
          (clés associées au mot de passe courant)
          [0] KERB_KEY_DATA_NEW.IterationCount 0x1000 (4096)
                               .KeyType: 0x12 (aes256-cts-hmac-sha1-96)
                               .Key: B67591D4092C3FF24A20F0997BF0C716 ↩
                                     2907CD49FFE78601F315D69EF0B71845
          [1] KERB_KEY_DATA_NEW.IterationCount 0x1000 (4096)
                               .KeyType: 0x11 (aes128-cts-hmac-sha1-96)
                               .Key: B897F17C374A35F620D6C4B8C79454AA
          [2] KERB_KEY_DATA_NEW.IterationCount 0x1000 (4096)
                               .KeyType: 0x3 (des-cbc-md5)
                               .Key: 76D3807CA8B60EC2
        KERB_STORED_CREDENTIAL_NEW.ServiceCredentialCount: 0
```

```
                              .OldCredentialCount: 3
      (clés associées au mot de passe n-1)
      [0] KERB_KEY_DATA_NEW.IterationCount 0x1000 (4096)
                           .KeyType: 0x12 (aes256-cts-hmac-sha1-96)
                           .Key: 3F5CDFE6194FC151CEFA92100973DD93 ↩
                                 DCB53014145717DF5751F539EDB0C6B5
      [1] KERB_KEY_DATA_NEW.IterationCount 0x1000 (4096)
                           .KeyType: 0x11 (aes128-cts-hmac-sha1-96)
                           .Key: D96943E817CB54FECA064958CC932188
      [2] KERB_KEY_DATA_NEW.IterationCount 0x1000 (4096)
                           .KeyType: 0x3 (des-cbc-md5)
                           .Key: 26CEEA54024ADAF2
    KERB_STORED_CREDENTIAL_NEW.OlderCredentialCount: 3
      (clés associées au mot de passe n-2)
      [0] KERB_KEY_DATA_NEW.IterationCount 0x1000 (4096)
                           .KeyType: 0x12 (aes256-cts-hmac-sha1-96)
                           .Key: 176E3A30AC2C18D02A108249C566C09D ↩
                                 2B62FDFFC83F92A3B5B5A8F106B4C62F
      [1] KERB_KEY_DATA_NEW.IterationCount 0x1000 (4096)
                           .KeyType: 0x11 (aes128-cts-hmac-sha1-96)
                           .Key: 5D0DAAECC642C13CF86398C26B36A46B
      [2] KERB_KEY_DATA_NEW.IterationCount 0x1000 (4096)
                           .KeyType: 0x3 (des-cbc-md5)
                           .Key: 43AE499E02C47C85
[1] USER_PROPERTY.PropertyName: Primary:Kerberos
    KERB_STORED_CREDENTIAL.Revision: 3
                          .DefaultSalt: "DEMO.TESTsstic"
                          .CredentialCount: 1
      (clé associée au mot de passe courant)
      [0] KERB_KEY_DATA.KeyType: 0x3 (des-cbc-md5)
                       .Key: 76D3807CA8B60EC2
    KERB_STORED_CREDENTIAL.OldCredentialCount: 1
      (clé associée au mot de passe n-1)
      [0] KERB_KEY_DATA.KeyType: 0x3 (des-cbc-md5)
                       .Key: 26CEEA54024ADAF2
[2] USER_PROPERTY.PropertyName: "Packages"
[3] USER_PROPERTY.PropertyName: "Primary:WDigest"
```

Listing 7. Données d'authentification associées à un compte utilisateur

2.4 Stockage de secrets dans les sessions d'authentification

Pour chaque utilisateur qui s'authentifie auprès d'un système, et ce quelle que soit la forme (interactivement, par le réseau, etc.), une session d'authentification est créée et contient diverses informations de contexte [22] (identifiant local unique de la session, nom de l'utilisateur, heure de connexion, etc.). En outre, les SSP [7] assurent la conservation de secrets d'authentification pour chaque session. Ceux-ci permettent, entre autres, de mettre en œuvre l'authentification implicite, c'est-à-dire

7. SSP : *Security Support Provider*. Composant en charge de l'authentification distante.

la possibilité pour les utilisateurs de s'authentifier à distance sans avoir à ressaisir leur mot de passe.

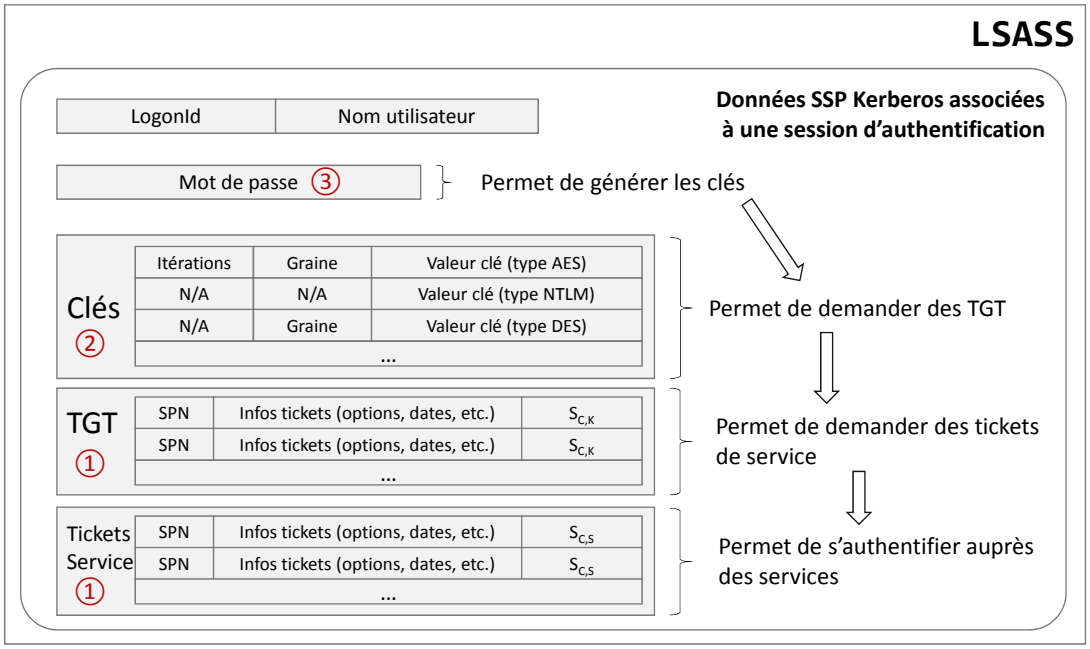

Figure 2. Données d'une session d'authentification Kerberos

Pour chaque session d'authentification, le SSP Kerberos sauvegarde, dans la mémoire du processus LSASS, des informations liées à l'authentification Kerberos (voir figure 2). Parmi celles-ci, on peut citer :

- les tickets (TGT ou ticket de service) issus des différents échanges avec le KDC (①) ;
- les clés dérivées depuis le mot de passe (②) : ces clés sont initialement calculées avec la graine par défaut. Plusieurs clés, de types différents (cf. partie 2.1), sont présentes afin d'être en mesure de supporter les algorithmes correspondants (`des-cbc-md5`, `aes128-cts-hmac-sha1-96`, etc.). Elles permettent de demander un TGT sans avoir à demander à l'utilisateur son mot de passe. Si ces clés n'étaient pas conservées en mémoire, l'utilisateur devrait ressaisir son mot de passe lors du renouvellement des TGT après leur expiration ;

– optionnellement, le mot de passe de l'utilisateur (③) : celui-ci est conservé en clair [8] jusqu'à réception d'un TGT.

La conservation en mémoire du mot de passe est nécessaire car, initialement, le client l'utilise pour calculer les clés avec les valeurs par défaut de la graine et du nombre d'itérations. Avant d'effacer le mot de passe, le client doit s'assurer que les valeurs utilisées correspondent à celles utilisées par le KDC, ce dernier ayant la possibilité d'utiliser une autre graine ou un nombre d'itérations différent. Dans les messages `KRB_AS_REP` (cf. partie 1.4), le KDC transmet au client la graine utilisée et le nombre d'itérations pour chaque type de clé via les structures `PA-PW-SALT` ou `PA-ETYPE-INFO2`. De même, si la préauthentification échoue, un message `KRB_ERROR` est envoyé du KDC au client avec ces mêmes structures. En cas de différences, le client peut alors recalculer les clés avec les bonnes valeurs. Lorsque le client est assuré que les clés sont correctement générées et fonctionnelles, le mot de passe est alors effacé.

Il est souvent reproché à Microsoft de conserver en mémoire des secrets liés aux mots de passe des utilisateurs authentifiés sur un système (empreinte NTLM, clés Kerberos, tickets Kerberos voire le mot de passe de l'utilisateur ou son code PIN). Ceci est rendu nécessaire pour les besoins du SSO [9] qui permet aux utilisateurs de s'authentifier sur le réseau sans devoir saisir à nouveau leur mot de passe. Il ne faut pas oublier que toutes les informations sensibles mises en cache le sont dans l'espace mémoire du processus LSASS et ne sont accessibles qu'à un attaquant ayant obtenu un haut niveau de privilège (c'est-à-dire administrateur local) sur un système.

Un effort de sécurisation doit toujours être réalisé pour éviter qu'un attaquant obtienne un haut niveau de privilège sur les postes de travail, d'administration ou sur les serveurs. Même si aucune information n'était mise en cache, un attaquant étant administrateur d'un système aurait d'autres possibilités pour récupérer le mot de passe de l'utilisateur (installation d'un *keylogger*, ajout de faux *Credential Providers* ou de faux SSP, etc.). Cependant, Microsoft a récemment mis en place des protections visant à limiter, plus ou moins, les possibilités d'attaques (*protected users*

8. Techniquement, toutes les informations sensibles sont conservées chiffrées dans la mémoire de LSASS. Ce chiffrement n'a qu'un rôle d'obfuscation, la clé de chiffrement étant également dans la mémoire du processus.

9. SSO : *Single sign-on*. Le terme authentification implicite est plus approprié aux environnements Windows.

group, *protected processes*, etc.). Ces protections sont détaillées dans la partie 11.

2.5 Autorisation

Kerberos, en tant que protocole d'authentification, n'est pas suffisant. En effet, le modèle du contrôle d'accès de Windows nécessite des informations supplémentaires relatives à l'UPN [10] d'un utilisateur authentifié.

Pour les systèmes Windows, l'autorisation est mise en œuvre par des mécanismes complexes, basés sur les jetons d'accès (*access token*), qui décrivent les contextes de sécurité des utilisateurs ainsi que les descripteurs de sécurité qui définissent les droits d'accès. Ces structures utilisent des SID (*Security Identifier*s) pour désigner des utilisateurs, groupes ou entités de sécurité.

Après l'authentification d'un client, un serveur doit générer un jeton d'accès afin de décrire le contexte de sécurité de celui-ci. Pour créer ce jeton, le serveur doit disposer, entre autres, de l'identifiant de sécurité de l'utilisateur et des identifiants de ses groupes d'appartenance. Or ce sont les KDC (c'est-à-dire les contrôleurs de domaine) qui disposent de ces informations.

Microsoft a donc défini une extension à Kerberos baptisée PAC (*Privilege Attribute Certificate*) qui permet aux contrôleurs de domaine de transmettre aux serveurs les informations nécessaires à l'établissement du contexte de sécurité du client et la génération de son jeton d'accès.

Pour ne pas modifier en profondeur le protocole Kerberos, Microsoft utilise le champ `authorization-data` des tickets. Ce champ est défini dans la norme Kerberos, mais son contenu est laissé à libre disposition pour le transfert de données d'autorisation.

Ainsi, lors de la demande d'un TGT par un client (traitement d'un message `KRB_AS_REQ`), le contrôleur de domaine initialise une structure `KERB_VALIDATION_INFO` et la remplit avec les informations d'autorisation relatives au client initiant la demande. Cette structure est définie dans [15] et les critères de remplissage des champs dans [12]. Un extrait de cette structure (qui compte plus de 35 champs) est donné ci-dessous :

10. UPN : *user principal name*. Nom unique d'identification d'un utilisateur dans un domaine Kerberos. Généralement de la forme `utilisateur@realm`.

```
typedef struct _KERB_VALIDATION_INFO {
  FILETIME PasswordLastSet;
  FILETIME PasswordCanChange;
  FILETIME PasswordMustChange;
  RPC_UNICODE_STRING FullName;
  RPC_UNICODE_STRING LogonScript;
  RPC_UNICODE_STRING ProfilePath;
  RPC_UNICODE_STRING HomeDirectory;
  USHORT LogonCount;
  USHORT BadPasswordCount;
  ULONG UserId;
  ULONG PrimaryGroupId;
  ULONG GroupCount;
  [size_is(GroupCount)] PGROUP_MEMBERSHIP GroupIds;
  PISID LogonDomainId;
  ULONG UserAccountControl;
  FILETIME LastSuccessfulILogon;
  FILETIME LastFailedILogon;
  ULONG FailedILogonCount;
  ULONG SidCount;
  [size_is(SidCount)] PKERB_SID_AND_ATTRIBUTES ExtraSids;
} KERB_VALIDATION_INFO;
```

Cette structure est directement dérivée de la structure `NETLOGON_VALIDATION_SAM_INFO4` [14] utilisée dans le protocole `NETLOGON` à des fins similaires (transfert du contexte de sécurité d'un utilisateur d'un contrôleur de domaine à un système validant une authentification d'un utilisateur).

Chaque nom de champ est plutôt explicite et décrit correctement le rôle associé, mais quelques précisions importantes peuvent être apportées :

- `LogonDomainId` contient le SID du domaine ;
- `UserId` contient le RID de l'utilisateur dans le domaine. Le SID de l'utilisateur est donc constitué à partir des champs `LogonDomainId` et `UserId` ;
- `GroupIds` contient les RID des groupes du domaine auxquels `GroupIds` appartient. Les SID de ces groupes sont donc constitués à partir des champs `LogonDomainId` et `UserId` ;
- `ExtraSids` contient les SID devant être ajoutés aux jetons des utilisateurs. Ceci est utilisé en particulier dans le mécanisme des *sIDHistory*.

Cette structure doit ensuite être doublement signée [11] :

11. Les algorithmes de signature sont choisis en fonction du type de clé.

- par la clé des KDC (K_{KDC}) : cette signature[12] est appelée la ***KDC Signature*** et est stockée dans une structure `PAC_SIGNATURE_DATA` ;
- par la clé du service destinataire du ticket : cette signature est appelée la ***Server Signature*** et est stockée dans une structure `PAC_SIGNATURE_DATA`. La clé du service destinataire peut être :
  - K_{KDC} dans le cas des TGT (le SPN étant `krbtgt/REALM@REALM`),
  - K_S dans le cas des tickets de service (le SPN étant typiquement `service/serveur@REALM`).

Toutes les structures sont encapsulées dans des structures de type `AD-WIN2K-PAC` et `AD-IF-RELEVANT` puis mises dans le champ `authorization-data` du ticket (figure 3, étape ①). Ce champ étant dans la partie chiffrée (`enc-part`), la PAC est ainsi chiffrée par la clé du service destinataire. Le listing 8 montre un exemple de ces structures dans un ticket.

Après leur réception, les TGT sont conservés en mémoire par les clients (cf. partie 2.4). Lors de la demande d'un ticket de service (via un message `KRB_TGS_REQ`), le KDC ne régénère pas une nouvelle PAC : celle-ci est recopiée depuis le TGT (figure 3, étape ②). Le processus de génération est le suivant :

1. le KDC déchiffre le TGT avec K_{KDC} et récupère la PAC ;
2. les signatures de la PAC sont vérifiées (*KDC Signature* et *Server Signature*) ;
3. le KDC génère un ticket pour le service demandé ;
4. la PAC récupérée du TGT est recopiée dans ce nouveau ticket ;
5. les deux signatures de la PAC sont générées :
   - *KDC Signature* est signée à nouveau par K_{KDC},
   - *Server Signature* est signée cette fois-ci par K_S ;
6. le ticket est chiffré avec la clé K_S du serveur destinataire.

```
[Ticket]
  tkt-vno: 5
  realm: DEMO.LOCAL
  sname: krbtgt/DEMO.LOCAL (2)
  enc-part: EncryptedData[EncTicketPart] etype: 18, kvno: 3
```

12. Le terme *signature*, bien qu'utilisé dans la documentation, est impropre s'agissant d'algorithmes à clés symétriques. Il serait préférable de parler de code d'authentification ou de HMAC.

```
[EncTicketPart] = Partie chiffrée =
  ...
  crealm: DEMO.TEST
  cname: client (1)
  ...
  *authorization-data:
    [AD-IF-RELEVANT]
      [AD-WIN2K-PAC]
        [PACTYPE] cBuffers: 3, Version: 0
          [PAC_INFO_BUFFER 0] ulType: 0x1, cbBufferSize: 456
            [KERB_VALIDATION_INFO]
              LogonTime: 2014/02/17 10:52:59.167
              PasswordLastSet: 2014/02/17 10:12:13.44
              PasswordCanChange: 2014/02/17 10:12:13.44
              PasswordMustChange: 2014/03/31 10:12:13.44
              EffectiveName: client
              FullName: client
              LogonCount: 54
              BadPasswordCount: 12
              UserId: 1112
              PrimaryGroupId: 513
              GroupIds (GroupCount: 1): 513 (7)
              UserFlags: 32
              LogonServer: DC-2012-01
              LogonDomainName: DEMOTEST
              LogonDomainId: S-1-5-21-235809208-238214944-1372333019
              UserAccountControl: 0x10
              ExtraSids (SidCount: 1): S-1-18-1 (7)
          [PAC_INFO_BUFFER 1] ulType: 0x6, cbBufferSize: 16
            [PAC_SIGNATURE_DATA] Server Signature
              SignatureType: 0x10 (HMAC_SHA1_96_AES256)
              Signature: da1495e631fbc0b814350194
          [PAC_INFO_BUFFER 2] ulType: 0x7, cbBufferSize: 20
            [PAC_SIGNATURE_DATA] KDC Signature
              SignatureType: 0xffffff76 (KERB_CHECKSUM_HMAC_MD5)
              Signature: 4525b9b08bef831a960e04b4819e6c9f
```

Listing 8. Structures liées à la PAC dans un ticket (extrait)

Quand un client présente un ticket de service (via un message `KRB_AP_REQ`), le serveur récupère la PAC dans la partie chiffrée du ticket et, après vérification des signatures, génère un jeton de sécurité qui représente le contexte de sécurité du client (figure 3, étape ③). Ce jeton servira alors dans le contrôle d'accès en particulier lorsque le client sera *impersonifié*[13]. Pour cela :

– le SID utilisateur (premier SID de la liste `UserAndGroups` d'un jeton) est construit à partir des champs `LogonDomainId` et `UserId` de la PAC ;

13. L'*impersonation* est le mécanisme qui consiste à emprunter l'identité d'un utilisateur dans le cadre du contrôle d'accès.

– les SID de groupes sont construits à partir des champs `LogonDomainId` et `GroupIds` et ajoutés dans la liste `UserAndGroups` ;
– tous les SID de la liste `ExtraSids` de la PAC sont ajoutés à la liste `UserAndGroups` du jeton.

Il est important de noter que le serveur, au sein d'un même domaine, fait une confiance absolue à la PAC générée par le KDC et contenue dans le ticket présenté.

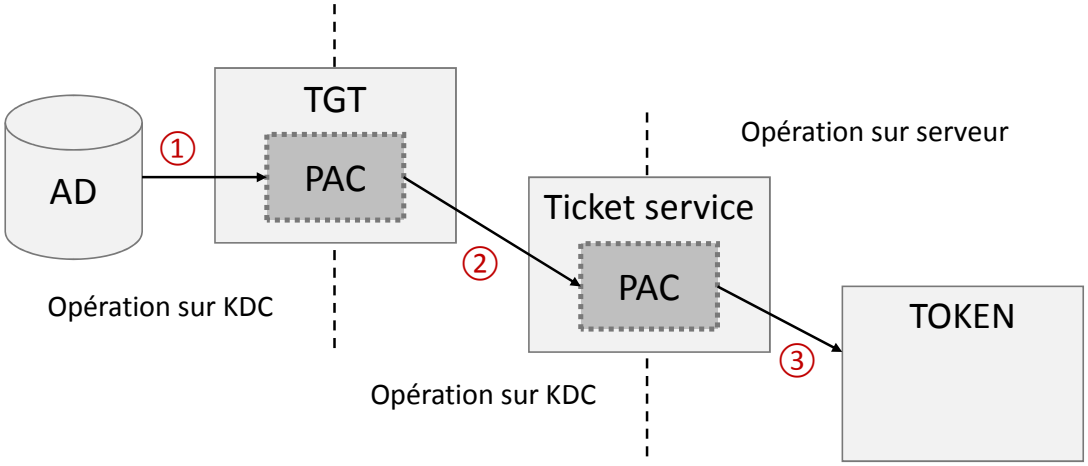

Figure 3. Synopsis de traitement d'une PAC

3 PKINIT

Dans sa version de base, Kerberos repose uniquement sur des algorithmes symétriques. L'utilisateur utilise des clés de chiffrement symétrique dérivées directement à partir de son mot de passe pour s'authentifier. Afin de permettre d'autres formes d'authentification, Microsoft a, dès Windows 2000, proposé une extension baptisée PKINIT. Celle-ci, standardisée depuis dans la RFC 4556, permet principalement la mise en œuvre de deux fonctionnalités :

– l'utilisation d'algorithmes asymétriques et de certificats X.509 pour l'authentification des utilisateurs ;
– l'utilisation d'algorithmes asymétriques ou de Diffie-Hellman pour l'échange ou la génération des clés de session.

Cette extension vise principalement à permettre l'utilisation de cartes à puce ou de jeton de sécurité matériel pour l'authentification des utilisateurs.

Les seuls changements par rapport à une authentification classique reposant sur un mot de passe et des clés symétriques concernent les échanges `KRB_AS_REQ` et `KRB_AS_REP` permettant à l'utilisateur d'obtenir un TGT et la clé de session associée. Une fois ce TGT obtenu, tous les autres échanges Kerberos sont identiques.

3.1 Échanges KRB_AS_REQ

Au niveau du message `KRB_AS_REQ` (listing 9), le client va indiquer le support de PKINIT et son souhait de s'authentifier via certificat en ajoutant un bloc de type `PA_PK_AS_REQ` dans la partie *preauthentication data* (`padata`) du message `KRB_AS_REQ`. Ce bloc contient, entre autres :

- un bloc `signedAuthPack` contenant le certificat du client, les certificats des autorités intermédiaires de la chaîne de certification (mais pas le certificat racine) ainsi qu'un bloc `AuthPack` signé avec la clé privée associée au certificat du client. Ce bloc contient :
  - un bloc de préauthentification (`pkAuthenticator`) contenant l'heure du client et un aléa (`nonce`),
  - optionnellement, la partie cliente de l'échange Diffie-Hellman ;
- optionnellement, un bloc `trustedCertifiers` contenant les certificats X.509 reconnus par le client pour authentifier le KDC.

```
[AS-REQ]
 pvno : numéro de version, toujours 5
 msg-type : type de message, 10 pour KRB_AS_REQ
 *padata: Séquence de blocs PA-DATA
   [PA-DATA]
     [PA-PK-AS-REQ]
       [signedAuthPack] Bloc signé par l'émetteur
         Certificat du signataire (usercert@demo.test)
         HashAlgorithm: 1.3.14.3.2.26 (szOID_OIWSEC_sha1)
         HashEncryptionAlgorithm: 1.2.840.113549.1.1.1 (szOID_RSA_RSA)
         [AuthPack]
           [pkAuthenticator]
             cusec: 536324
             ctime: 2014/02/17 10:58:37.0
             nonce: 0x349eb58e
 req-body:
   [KDC-REQ-BODY]
     kdc-options: 0x40810010 (enc_pa_rep, renewable, forwardable)
     *cname: usercert@demo.test (10)
     ...
```

Listing 9. Message `KRB_AS_REQ` avec extension PKINIT

3.2 Validation par l'Active Directory

En traitant le bloc `signedAuthPack` émis par le client, le KDC récupère le certificat de l'utilisateur qu'il doit ensuite associer à un compte de l'Active Directory. Pour les utilisateurs, de manière générale, le champ *Subject Alternative Name* du certificat doit correspondre à l'UPN d'un utilisateur dans l'annuaire, soit sous forme implicite (`samAccountName@domain_FQDN`), soit sous forme explicite (attribut `userPrincipalName` de l'objet utilisateur). Le processus complet d'association et ses subtilités est décrit dans la documentation de PKINIT [19].

En outre, le certificat utilisateur doit être signé par un certificat d'autorité (racine ou intermédiaire) autorisé pour l'authentification par carte à puce. Pour cela, le certificat de l'autorité doit être préalablement ajouté à l'attribut `cACertificate` du conteneur `NTAuthCertificates` [14].

3.3 Échanges KRB_AS_REP

Dans le cadre d'une authentification PKINIT, les messages de réponse du KDC (`KRB_AS_REP`, listing 10) sont également étendus afin d'ajouter un bloc `PA-PK-AS-REP` dans la partie *preauthentication data* (`padata`) du message. Ce bloc contient, au choix :

- un bloc `dhInfo` contenant la partie serveur de l'échange Diffie-Hellman signée par la clé privée associée au certificat du KDC (afin d'authentifier l'échange) ;
- un bloc `encKeyPack` contenant la clé de session générée par le KDC et chiffrée par la clé publique du certificat de l'utilisateur.

Dans les deux cas, le client récupère une clé qui permet le déchiffrement de la partie chiffrée de la réponse `KRB_AS_REP`. Pour rappel, dans cette partie, le champ `EncryptionKey` contient la clé de session $S_{C,K}$ associée au TGT. La récupération de cette clé et la possession du TGT prouveront, pour la suite, l'authenticité du client.

```
[AS-REP]
  pvno : numéro de version, toujours 5
  msg-type : type de message, 11 pour KRB_AS_REP
  *padata: Séquence de blocs PA-DATA
    [PA-PK-AS-REP] Choix encKeyPack
```

14. `CN=NTAuthCertificates,CN=Public Key Services,CN=Services,-CN=Configuration,DC=domain,DC=tld`

```
    [ReplyKeyPack] == Partie chiffrée ==
    replyKey: [EncryptionKey] keytype: 0x12, keyvalue: S_AS-REP
    asChecksum:
      [Checksum]
        cksumtype: 0xe
        checksum: 83390173a571ac053be08789838a61a8f79f43ff
  crealm: DEMO.TEST
  cname: usercert (1)
  ticket:
    [Ticket] TGT transmis au client
      tkt-vno: numéro de version, toujours 5
      realm: DEMO.TEST
      sname: krbtgt/DEMO.TEST (2)
      enc-part: EncryptedData[EncTicketPart] etype: 18, kvno: 2
        [EncTicketPart] == Partie chiffrée par K_KDC ==
        ...
        key: [EncryptionKey] keytype: 0x12, keyvalue:
          Clé de session S_C,K contenue dans le ticket
  enc-part: EncryptedData[EncASRepPart] etype: 0x12
    [EncKDCRepPart] == Partie chiffrée par S_AS-REP ==
    key: [EncryptionKey] keytype: 0x12, keyvalue:
      Clé de session S_C,K transmise au client
    ...
```

Listing 10. Message `KRB_AS_REP` avec extension PKINIT

À partir de cet instant, la suite du processus d'authentification est identique à une authentification Kerberos classique : au moyen du TGT, le client peut demander des tickets de service afin de s'authentifier auprès de serveurs ou services.

3.4 Le retour de NTLMSSP

Microsoft utilise également PKINIT pour une fonctionnalité supplémentaire propre aux environnements Windows. De base, PKINIT permet aux utilisateurs de s'authentifier via un protocole asymétrique lors de la demande d'un TGT. Celui-ci permet ensuite d'obtenir des tickets de service et de s'authentifier, toujours via Kerberos, auprès de serveurs ou de services. Dans cette configuration, l'utilisateur n'ayant pas saisi son mot de passe (qu'il peut ne pas connaître), le SSP `msv1_0`[15] ne peut générer les empreintes LM et NTLM de l'utilisateur. Celles-ci ne sont donc pas présentes dans sa session d'authentification ce qui rend impossible l'authentification implicite via les protocoles NTLM.

15. Ce SSP est en charge de l'authentification via les protocoles LM, NTLM, NTLMv2, etc.

Or les contrôleurs de domaine conservent, dans l'Active Directory, les empreintes NTLM des utilisateurs. Celles-ci peuvent correspondre à celles d'un mot de passe choisi par l'utilisateur ou d'un mot de passe généré automatiquement si l'authentification par carte à puce est forcée (cf. partie 4.2). Ainsi, pour que l'utilisateur puisse également s'authentifier en NTLM, PKINIT permet aux contrôleurs de domaine de transférer les empreintes NTLM aux clients lorsque ceux-ci s'authentifient avec les extensions PKINIT.

Le listing 11 détaille une structure `PAC_CREDENTIAL_INFO` dans un message `KRB_AS_REP`.

```
[Ticket]
  tkt-vno: 5
  realm: DEMO.LOCAL
  sname: krbtgt/DEMO.LOCAL (2)
  enc-part: EncryptedData[EncTicketPart] etype: 18, kvno: 3
  [EncTicketPart] == Partie chiffrée ==
    ...
    crealm: DEMO.TEST
    cname: client (1)
    ...
    *authorization-data:
      [AD-IF-RELEVANT]
        [AD-WIN2K-PAC]
          [PACTYPE] cBuffers: 3, Version: 0
            [PAC_INFO_BUFFER 0] ulType: 0x1, cbBufferSize: 456
            ...
            [PAC_INFO_BUFFER 1] ulType: 0x6, cbBufferSize: 16
            ...
            [PAC_INFO_BUFFER 2] ulType: 0x7, cbBufferSize: 20
            ...
            [PAC_INFO_BUFFER 3] ulType: 0x2, cbBufferSize: 148
              [PAC_CREDENTIAL_INFO] Version: 0, EncryptionType: 12
                [PAC_CREDENTIAL_DATA] CredentialCount: 1
                  [SECPKG_SUPPLEMENTAL_CRED] PackageName: NTLM
                    [NTLM_SUPPLEMENTAL_CREDENTIAL]
                      Version: 0, Flags: 0x2
                      LmPassword: 00000000000000000000000000000000
                      NtPassword: 95771430787a7b69eacad46660d7ff06
```

Listing 11. Structure `PAC_CREDENTIAL_INFO`

4 Mythes autour de Kerberos

4.1 Protection contre les attaques de type Pass-the-hash

Les attaques de type *Pass-the-hash* consistent à s'authentifier, non pas à l'aide du mot de passe d'un utilisateur, mais uniquement au moyen

de l'empreinte de son mot de passe. De manière générale, les protocoles d'authentification à distance basés sur des empreintes sans graine sont vulnérables à ce type d'attaque, ce qui est le cas du protocole NTLM et de toutes ses variantes (LM, NTLMv2, etc.).

Si un attaquant récupère une base de comptes (une base SAM ou une base d'un Active Directory), il peut alors s'authentifier à distance avec les empreintes sans avoir à retrouver les mots de passe en clair.

Kerberos souffre de la même problématique : si un attaquant arrive à extraire les clés Kerberos (cf. partie 2.4), il est alors en mesure de demander un TGT sans connaître le mot de passe associé. En effet, pour chiffrer le bloc de préauthentification (`PA-ENC-TIMESTAMP`) ou pour déchiffrer la clé de session associée au TGT, seule la clé K_C du client (connue de celui-ci et également stockée dans l'Active Directory [16]) est nécessaire.

Sous Windows, ce type d'attaque peut aisément être mis en œuvre en remplaçant, dans la mémoire de LSASS, les clés Kerberos d'une session d'authentification donnée. Après substitution des clés et effacement de tous les tickets du cache (`klist purge`), toute demande d'authentification implicite basée sur Kerberos déclenchera automatiquement une demande de TGT en utilisant les nouvelles clés injectées. On peut alors parler d'attaque de type *Pass-the-keys*.

Il ne s'agit cependant pas d'une vulnérabilité, mais du mode de fonctionnement inhérent à Kerberos et qui touche tous les systèmes (Windows et l'Active Directory n'étant pas les seuls concernés). Ainsi, dans toute architecture qui met en œuvre Kerberos, la protection des bases des clés des KDC doit être une priorité absolue.

4.2 Utilisation des cartes à puce

L'utilisation des cartes à puce est souvent présentée comme une solution à de nombreux problèmes liés à l'authentification. Or, comme décrit dans la partie 3, l'authentification par carte à puce apporte uniquement, dans le contexte de Kerberos, une nouvelle méthode de préauthentification et de récupération de la clé de session pour un

16. Comme vu dans la partie 2.1, un client a plusieurs types de clés K_C, mais le principe est identique pour chaque type de clé.

utilisateur lors de la demande de son TGT.

Une option intéressante consiste à « forcer » l'utilisation de la carte à puce en sélectionnant l'option « Une carte à puce est nécessaire pour ouvrir une session interactive » dans les propriétés d'un compte utilisateur. Cela revient à positionner l'option `ADS_UF_SMARTCARD_REQUIRED` de l'attribut userAccountControl [23] sur le compte.

Lorsque cette option est activée, le comportement des composants d'authentification est ainsi modifié :

- le SSP msv1_0 interdit toutes les formes d'authentification, exceptées celles du type `LOGON32_LOGON_NETWORK` correspondant aux authentifications réseau ou à distance. Ainsi, un utilisateur ne peut plus s'authentifier interactivement avec son mot de passe, mais un attaquant peut toujours continuer à s'authentifier à distance avec l'empreinte NTLM via une attaque de type *Pass-the-hash* ;
- le SSP Kerberos délivre des TGT via le service AS uniquement si une extension PKINIT (cf. partie 3) est présente dans la requête de l'utilisateur. Les autres services (TGS et AP) ne sont pas affectés.

L'option `ADS_UF_SMARTCARD_REQUIRED` impose donc à un utilisateur de s'authentifier interactivement via un dispositif prenant en charge PKINIT et Kerberos. Concernant les authentifications réseau, si le protocole NTLM est toujours activé, les attaques de type *Pass-the-hash* sont encore possibles via NTLM. En revanche, si NTLM est désactivé [16], un attaquant qui aurait récupéré les secrets d'un utilisateur (empreinte NTLM ou clés Kerberos) ne pourra pas demander un TGT via une attaque de type *Pass-the-key*, ce qui rend l'authentification par Kerberos impossible. Dans tous les cas, l'option ne permet pas de se prémunir contre le vol d'un TGT et de la clé de session : le TGT dérobé peut toujours être utilisé par l'attaquant pour demander un ticket de service et usurper l'identité de l'utilisateur. Cependant, la durée de vie des tickets limite, dans le temps, la faisabilité de l'attaque.

Afin d'optimiser le gain de sécurité apporté par la carte à puce dans le contexte de Kerberos, il convient de forcer la carte à puce pour les utilisateurs et de désactiver l'authentification NTLM (ce qui est actuellement compliqué à configurer). À défaut, le gain en sécurité restera limité.

5 Génération de PAC dans les tickets

5.1 Principe général

Comme décrit dans la partie 2.5, dans un environnement Active Directory, Kerberos est utilisé pour l'authentification ainsi que pour l'autorisation via le mécanisme de la PAC qui permet aux contrôleurs de domaine de transmettre, via une structure `KERB_VALIDATION_INFO`, les données d'autorisation.

Si un attaquant dispose de certaines clés présentes dans l'Active Directory, il est en mesure de signer correctement des PAC et de chiffrer les parties `enc-part` des tickets (TGT ou ticket de service). Cela lui confère donc la possibilité de générer des tickets de toutes pièces avec une PAC valide.

Pour pouvoir générer un ticket de toutes pièces avec une PAC valide, deux clés sont nécessaires (cf. partie 2.5) :
- $\mathtt{K}_{\mathtt{KDC}}$ pour générer la signature *KDC Signature* ;
- la clé du service destinataire du ticket pour :
  - générer la signature *Server Signature*,
  - chiffrer la partie `enc-part` du ticket.

5.2 Génération de TGT

Les TGT sont des tickets au même titre que les tickets de service avec quelques particularités, parmi lesquelles leur SPN qui est de la forme `krbtgt/REALM@REALM`. Le service destinataire étant le TGS, la clé du service destinataire est $\mathtt{K}_{\mathtt{KDC}}$.

La seule connaissance de $\mathtt{K}_{\mathtt{KDC}}$ est donc nécessaire pour pouvoir forger un TGT avec une PAC et permet :
- de générer les deux signatures de la PAC (*KDC Signature* et *Server Signature*) ;
- de chiffrer la partie `enc-part` du ticket.

Avec un ticket TGT forgé de toutes pièces, il devient possible de s'authentifier auprès du service TGS d'un KDC pour demander des tickets de service. La PAC contenue dans le TGT sera déchiffrée, vérifiée et recopiée telle quelle dans le ticket de service généré.

Ainsi, la connaissance de K_{KDC} permet de forger une PAC et un TGT puis d'obtenir un ticket de service pour n'importe quel serveur ou service du royaume Kerberos (c'est-à-dire du domaine Active Directory). La compromission du compte `krbtgt` offre donc à un attaquant une emprise complète sur le domaine.

5.3 Génération de tickets de service

Pour pouvoir forger un ticket de service, deux clés sont a priori nécessaires :

- K_{KDC} pour générer la signature *KDC Signature* de la PAC ;
- la clé du service destinataire pour générer la signature *Server Signature* de la PAC et chiffrer la partie `enc-part` du ticket.

Comme vu dans la partie 2.3, au sein de l'Active Directory, les clés des services sont généralement celles du serveur qui les héberge. Par exemple, pour le SPN `cifs/serveur-01.demo.test`, K_S correspond aux clés du compte machine `serveur-01` dans l'Active Directory.

Si on perçoit bien l'utilité de la signature de la PAC avec K_S (pour générer la signature *Server Signature*) ainsi que le chiffrement du ticket, on peut légitimement se questionner sur l'utilité de la signature de la PAC avec K_{KDC} (pour générer la signature *KDC Signature*) puisque le serveur destinataire ne connaît pas cette clé et ne peut donc pas vérifier cette signature.

Cette signature est utilisée dans le cadre de la « validation de la PAC » [5]. Ce mécanisme permet au serveur de valider la PAC auprès du KDC via le protocole `NETLOGON`. Dans le cas où la validation de la PAC est activée (ce qui n'est pas systématique), le serveur retransmet la PAC reçue et la *KDC Signature* à un KDC qui doit valider la signature avec K_{KDC}.

Cette vérification entraîne une latence réseau importante ainsi qu'une communication entre le serveur et le KDC. Dans les faits, elle est rarement utilisée [30], voire désactivée par défaut depuis Windows Server 2008 [25].

Ainsi, dans la majorité des cas, seule la clé K_S est nécessaire pour générer un ticket de service valide. Si un attaquant dispose des clés associées à un compte machine, il est donc en mesure de générer des tickets de service pour cette machine avec une PAC arbitraire et ainsi d'y accéder avec n'importe quel droit ou privilège.

5.4 Scénarios envisageables

Comme démontré dans les paragraphes précédents, si un attaquant dispose des clés du compte `krbtgt` ou d'un compte de serveur, il est en mesure de demander ou de générer des tickets de service avec une PAC modifiée à sa guise.

Plusieurs scénarios sont alors envisageables, mais le plus intéressant consiste à présenter un ticket au nom d'un utilisateur quelconque, tout en ajoutant des SID de groupes auxquels l'utilisateur n'appartient pas. Ainsi, si des SID de groupes sont frauduleusement ajoutés, l'utilisateur authentifié va disposer de droits d'accès et de privilèges supplémentaires lui permettant d'accéder à des ressources comme s'il faisait légitiment partie de ces groupes. Parmi les SID les plus intéressants à ajouter, on peut citer :

- le `SID S-1-5-21-domaine-512` correspondant au groupe « *Administrateurs du domaine* » disposant de droits et privilèges sur l'ensemble des ressources gérées par l'Active Directory ;
- le SID `S-1-5-18` correspondant à l'entité SYSTEM et conférant le niveau de droit maximal auprès du système Windows.

Il s'agit donc d'un moyen très aisé de réaliser des élévations de privilèges. Ceci montre l'importance de mettre en place une journalisation et un processus de traitement adéquat (cf. partie 10) afin d'être en mesure de détecter toute anomalie.

6 Relations interdomaines

Une fonctionnalité importante de Kerberos est la possibilité d'établir des relations de confiance (ou relations d'approbation) entre deux domaines Kerberos (*cross-realm trust*). Ce mécanisme, repris par Microsoft dans l'Active Directory, a permis de simplifier les relations entre domaines Windows et de faire apparaître la notion de forêt.

6.1 Relations du point de vue Kerberos

Lorsqu'un domaine Kerberos approuve un autre domaine, un compte (appelé compte de *trust*) symbolisant la relation est créé dans chaque domaine et un secret, partagé entre les deux domaines, y est associé. Ce secret permet la création de *referral tickets* mis en œuvre lorsqu'un utilisateur d'un domaine *approuvant* souhaite s'authentifier sur un

domaine *approuvé*.

Afin d'illustrer le processus d'authentification entre deux domaines, prenons l'exemple d'un utilisateur d'un domaine A (`userA @ DOMAINEA`) qui souhaite s'authentifier sur un service d'un domaine B (`serviceB/serveurB @ DOMAINEB`). Le processus d'échange de tickets Kerberos est décrit ci-dessous.

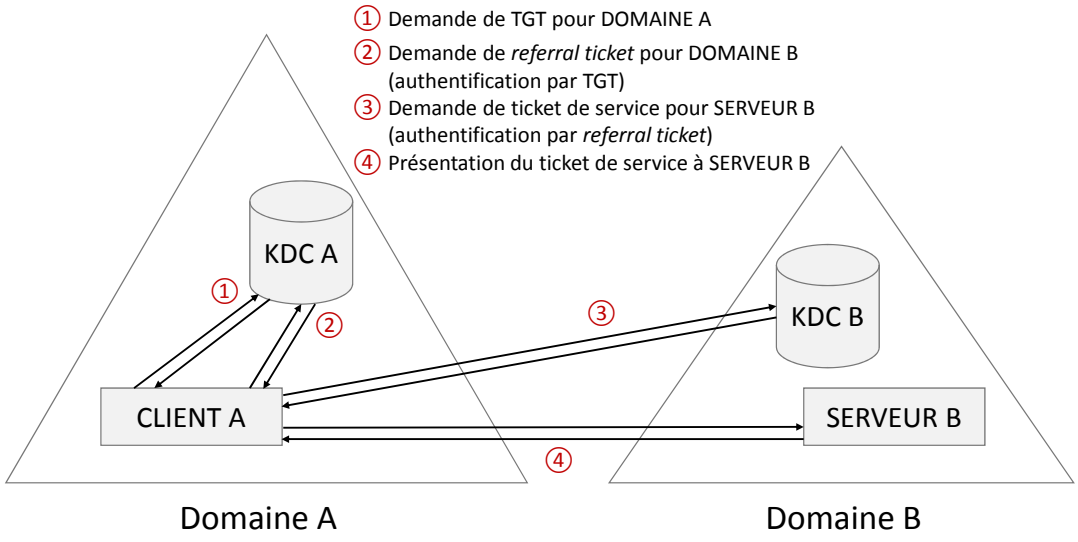

Figure 4. Synopsis d'une authentification *cross-domain*

L'utilisateur `userA` commence par récupérer un TGT auprès d'un KDC de son domaine via un échange `KRB_AS_REQ`/`KRB_AS_REP` (figure 4, étape ①) :

- `[KRB_AS_REQ]` - Demande de TGT :
  - demandeur : `userA @ DOMAINE A` ,
  - service demandé : `krbtgt/DOMAINEA @ DOMAINEA` ;
- `[KRB_AS_REP]` - Fourniture d'un TGT (la partie `enc-part` du TGT est chiffrée au moyen de la clé K_{KDC} du domaine A) :
  - pour : `userA @ DOMAINEA` ;
  - service : `krbtgt/DOMAINEA @ DOMAINEA`.

L'utilisateur demande ensuite un *referral ticket* pour le domaine B auprès d'un KDC du domaine A. Cette demande prend la forme d'une demande classique de ticket (échange `KRB_TGS_REQ` / `KRB_TGS_REP`) vers un KDC de son domaine (figure 4, étape ②) :

- [KRB_TGS_REQ] - Demande de *referral ticket* pour le domaine B :
  - partie PA-TGS-REQ : fourniture du TGT précédemment reçu,
  - service demandé : krbtgt/DOMAINEB @ DOMAINEB ;
- [KRB_TGS_REP] - Fourniture d'un *referral ticket* pour le domaine B :
  - Pour : userA @ DOMAINEA ;
  - service : krbtgt/DOMAINEB @ DOMAINEB.

La spécificité concerne la partie (enc-part) du *referral ticket* généré : celle-ci est chiffrée avec la clé du compte de *trust* entre le domaine A et le domaine B.

Le *referral ticket* tient lieu de TGT pour le domaine B et permet ensuite à l'utilisateur du domaine A de s'authentifier auprès d'un KDC du domaine B en vue d'obtenir des tickets de service pour le domaine B. Le client demande alors un ticket de service pour un serveur du domaine B. Pour cela, il s'adresse à un KDC du domaine B et présente le *referral ticket* précédemment obtenu (figure 4, étape ③) :

- [KRB_TGS_REQ] - Demande de ticket de service :
  - partie PA-TGS-REQ : fourniture du *referral ticket*.
  - service demandé : serviceB/serveurB @ DOMAINEB ;
- [KRB_TGS_REP] - Fourniture d'un ticket de service :
  - pour : userA @ DOMAINEA,
  - service : service/serveur @ DOMAINEB.

Le *referral ticket* étant chiffré par les clés associées au compte du *trust*, cela permet au KDC du domaine B d'authentifier le client. Le KDC délivre alors un ticket de service pour son domaine et le client n'a plus qu'à le présenter via un échange KRB_AP_REQ / KRB_AP_REP au service du domaine B (figure 4, étape ④).

6.2 Relations du point de vue Active Directory

Au sein d'un domaine Active Directory, une relation entre domaines se concrétise par un objet de classe trustedDomain symbolisant le *trust* (on parle alors de TDO pour *Trusted Domain Object*). Celui-ci est créé dans le conteneur System de l'annuaire du domaine. Par exemple, l'objet suivant définit une relation d'approbation entre le domaine domaineA.test et le domaine domaineB.test.

```
Dn: CN=domaineB.test,CN=System,DC=domaineA,DC=test
cn: domaineB.test;
flatName: DOMAINEB;
name: domaineB.test;
```

```
objectCategory: CN=Trusted-Domain,CN=Schema,CN=Configuration,DC=domaineA,DC=test;
objectClass (3): top; leaf; trustedDomain;
securityIdentifier: <ldp: Binary blob 24 bytes>;
trustAttributes: 0x20 = ( WITHIN_FOREST );
trustDirection: 3 = ( BIDIRECTIONAL );
trustPartner: domaineB.test;
trustType: 2 = ( UPLEVEL );
```

Tous les attributs d'un objet TDO sont décrits dans [9]. Les différents types de relations d'approbation, définis par les attributs `trustAttributes`, `trustType` et `trustDirection` sont détaillés dans la partie 6.4.

Dans l'exemple précédent, les attributs `trustAuthIncoming` et `trustAuthOutgoing` qui hébergent les secrets liés à la relation d'approbation n'apparaissent pas (ceux-ci étant de type `SecretAttribut` [8]).

Par compatibilité avec les relations d'approbation des domaines Windows NT 4 et les *trusts* mettant en œuvre `NETLOGON`, un autre objet est également créé. Celui-ci correspond à un compte utilisateur, mais de type `TRUST_ACCOUNT`.

```
Dn: CN= DOMAINEB$,CN=Users,DC=domaineA,DC=test
name: DOMAINEB$;
objectCategory: CN=Person,CN=Schema,CN=Configuration,DC=domaineA,DC=test;
objectClass (4): top; person; organizationalPerson; user;
objectSid: S-1-5-21-3839532325-4247084556-223454071-3104;
primaryGroupID: 513 = ( GROUP_RID_USERS );
sAMAccountName: DOMAINEB$;
sAMAccountType: 805306370 = ( TRUST_ACCOUNT );
userAccountControl: 0x820 = ( PASSWD_NOTREQD | INTERDOMAIN_TRUST_ACCOUNT );
```

Là encore, les attributs stockant les secrets d'authentification n'apparaissent pas (`unicodePwd`, `supplementalCredentials`, etc.).

6.3 PAC dans les relations d'approbation

Un aspect fondamental de la sécurité des relations d'approbation est le traitement de la PAC au sein des *referral tickets*.

Si l'on reprend l'exemple de l'authentification d'un utilisateur du domaine A auprès d'un service du domaine B (cf. partie 6.1), le traitement de la PAC est le suivant : lors de la génération du TGT par un KDC du domaine A pour l'utilisateur `userA @ DOMAINEA`, une PAC représentant

le contexte de sécurité de cet utilisateur est générée par le contrôleur de domaine et contient (à titre d'exemple) :

- `LogonDomainId` : SID du domaine A ;
- `UserId` : RID de l'utilisateur dans le domaine A ;
- `GroupIds` : liste des identifiants des groupes (RID) d'appartenance de l'utilisateur dans le domaine A ;
- `ExtraSids` : Liste de SID supplémentaires ajoutés, en particulier ceux présents dans le champ `sIDHistory` ou S-1-18-1 [17].

Lors de la génération du *referral ticket*, la PAC est recopiée depuis le TGT vers le *referral ticket* puis est chiffrée avec le secret du compte du *trust*.

Lorsque le KDC du domaine B est sollicité pour générer un ticket de service pour l'utilisateur du domaine A, il se base sur la PAC contenue dans le *referral ticket* présenté pour générer la PAC du ticket de service. Si le KDC du domaine B avait une confiance absolue envers le KDC du domaine A (c'est-à-dire si la PAC était recopiée telle quelle du *referral ticket* vers le ticket de service), cela présenterait un risque important de sécurité pour le domaine B. En effet, si le domaine A venait à être compromis, un attaquant serait en mesure de d'augmenter ses droits auprès du domaine B en rajoutant arbitrairement des SID dans la PAC du *referral ticket*. Des SID intéressants à rajouter seraient par exemple celui du groupe « *Administrateurs du domaine* » du domaine B ou celui de l'entité SYSTEM (S-1-5-18).

Cet ajout de SID peut être effectué de deux manières différentes :

- en éditant l'attribut `sIDHistory` d'un utilisateur : même si cette opération n'est pas triviale, elle reste possible pour quelqu'un ayant gagné des privilèges d'administration sur un contrôleur de domaine ;
- en forgeant de toutes pièces une PAC dans un *referral ticket* : cette méthode repose sur le même principe que celui décrit dans la partie relatives à la génération de TGT (cf. partie 5.2). La seule différence concerne la clé nécessaire au chiffrement du ticket et la signature de la PAC : il ne s'agit plus de celle du compte `krbtgt`, mais de celle

17. Apparu avec Windows Server 2012, ce SID baptisé `AUTHENTICATION_AUTHORITY_ASSERTED_IDENTITY` est inséré lorsque l'utilisateur s'est authentifié avec ses *credentials* (*crédentités*). Il est à opposer au SID `SERVICE_ASSERTED_IDENTITY` (S-1-18-2) inséré lorsqu'un service prend l'identité d'un utilisateur sans authentification de ce dernier (via le mécanisme *S4U2Self*, voir partie 8.1).

du compte du *trust*.

La récupération par un attaquant des secrets des comptes d'un *trust* d'un domaine met donc en péril les domaines approuvant celui-ci au travers d'une relation d'approbation.

Afin de se prémunir contre de tels scénarios, les contrôleurs de domaine Active Directory appliquent systématiquement une politique de filtrage des SID lorsqu'ils traitent un *referral ticket*. Cette politique permet de déterminer les SID qui sont conservés (c'est-à-dire ceux recopiés dans le ticket de service) et ceux qui sont filtrés donc exclus. La politique appliquée dépend du type de la relation d'approbation existante entre les deux domaines. La partie suivante détaille les différents types de relations d'approbation et la partie 6.5 détaille les différentes politiques de filtrage de SID.

6.4 Types de relations d'approbation

Le type d'une relation d'approbation entre deux domaines est principalement caractérisé par trois attributs du TDO de la relation d'approbation :

- `trustDirection` qui spécifie le sens de la relation ;
- `trustType` qui spécifie le type de la relation ;
- `trustAttributes` qui spécifie divers attributs et propriétés de la relation.

L'attribut `trustDirection` indique le sens de la relation d'approbation, donc celui de la confiance, et peut prendre les valeurs suivantes :

- `TRUST_DIRECTION_INBOUND` : indique que la relation d'approbation est unidirectionnelle dans le sens entrant, c'est-à-dire que le domaine est approuvé par le domaine distant ;
- `TRUST_DIRECTION_OUTBOUND` : indique que la relation d'approbation est unidirectionnelle dans le sens sortant, c'est-à-dire que le domaine approuve le domaine distant ;
- `TRUST_DIRECTION_BIDIRECTIONAL` : indique que la relation d'approbation est bidirectionnelle.

Si l'on se place dans un domaine donné, ce sont donc les valeurs `OUTBOUND` et `BIDIRECTIONAL` qui sont potentiellement dangereuses, car elles indiquent que le domaine concerné approuve un domaine distant, c'est-à-dire que les *referral tickets* en provenance de ce domaine sont

acceptés.

L'attribut `trustType` indique le type de relation et peut prendre les valeurs suivantes :

- `TRUST_TYPE_DOWNLEVEL` : indique que le domaine distant n'est pas de type Active Directory, c'est-à-dire qu'il s'agit d'une relation avec un domaine de type Windows NT ;
- `TRUST_TYPE_UPLEVEL` : indique que le domaine distant est de type Active Directory ;
- `TRUST_TYPE_MIT` : indique que le domaine distant n'est ni de type Active Directory ni Windows NT. Il s'agit généralement d'un domaine Kerberos de type MIT ou Heimdal. Dans ce cas, l'autorisation n'est pas prise en compte et les éventuelles PAC des *referral tickets* sont rejetées.

Enfin, l'attribut `trustAttributes` indique les propriétés de la relation d'approbation. Les principales propriétés sont :

- `TRUST_ATTRIBUTE_WITHIN_FOREST` ;
- `TRUST_ATTRIBUTE_FOREST_TRANSITIVE` ;
- `TRUST_ATTRIBUTE_CROSS_ORGANIZATION` ;
- `TRUST_ATTRIBUTE_CROSS_ORGANIZATION_NO_TGT_DELEGATION` ;
- `TRUST_ATTRIBUTE_QUARANTINED_DOMAIN` ;
- `TRUST_ATTRIBUTE_TREAT_AS_EXTERNAL`.

Une des propriétés les plus importantes est `WITHIN_FOREST` qui détermine si les deux domaines appartiennent à la même forêt. Cette propriété est positionnée automatiquement pour les relations intraforêt de type *Parent-child trust* ou *Tree-root trust*.

Si `WITHIN_FOREST` n'est pas positionné, il s'agit d'une relation extraforêt. Dans ce cas, les caractéristiques de la relation peuvent être précisées au moyen des propriétés suivantes :

- `FOREST_TRANSITIVE` : indique qu'il s'agit d'une relation entre deux forêts par opposition à une relation entre deux domaines. Cela influe sur la transitivité, le sens des relations et la politique de filtrage ;
- `CROSS_ORGANIZATION` : indique que les deux domaines liés par la relation d'approbation sont de deux organisations différentes. Dans ce cas, le SID `OTHER_ORGANIZATION` (S-1-5-1000) est ajouté dans la PAC lors du traitement d'un *referral ticket*. Ce SID peut être

utilisé pour interdire l'accès aux utilisateurs extérieurs à la forêt (c'est-à-dire à l'organisation) ;
- `TREAT_AS_EXTERNAL` indique, uniquement dans le cas où `FOREST_TRANSITIVE` est positionné, que la relation est de type « externe » (ce qui va influer sur la politique de filtrage des SID) ;
- `QUARANTINED_DOMAIN` va modifier la politique de filtrage de SID. Cette propriété peut être modifiée au moyen de la commande `netdom trust /quarantine` [6].

Les relations de type `DOWNLEVEL` sont toujours considérées comme des relations de type extraforêt et la propriété `TRUST_ATTRIBUTE_CROSS_ORGANIZATION_NO_TGT_DELEGATION` permet de désactiver entièrement la délégation sur un domaine approuvé en ne positionnant jamais l'option `ok-as-delegate` dans les tickets de service émis.

6.5 Filtrage des SID

Les différentes politiques de filtrage des PAC sont détaillées dans [15]. La détermination de la politique dépend des propriétés de la relation d'approbation. Les différentes politiques sont :
- ***WithinForest*** : appliquée à une relation ayant la propriété `WITHIN_FOREST` ;
- ***QuarantinedWithinForest*** : appliquée à une relation ayant les propriétés `WITHIN_FOREST` et `QUARANTINED_DOMAIN` activées ;
- ***CrossForest*** : appliquée à une relation ayant la propriété `FOREST_TRANSITIVE` ;
- ***External*** : appliquée à une relation (au choix) :
  - n'ayant pas la propriété `FOREST_TRANSITIVE`,
  - ayant les propriétés `FOREST_TRANSITIVE` et `TREAT_AS_EXTERNAL` activées simultanément ;
- ***QuarantinedExternal*** : appliquée à une relation ayant la propriété `QUARANTINED_DOMAIN`, mais pas `WITHIN_FOREST`.

Chaque SID présent dans la PAC du *referral ticket* est catégorisé en différentes catégories suivant les critères suivants :
- ***AlwaysFilter*** : SID systématiquement filtré. Parmi ces SID, on peut citer ceux sensibles du point de vue de la sécurité tels que `S-1-5-18` (SYSTEM), `S-1-5-32-544` (Administrateurs), `S-1-5-32-549` (Opérateurs de serveurs), etc. ;

- ***NeverFilter*** : SID jamais filtré. Il s'agit de SID très particuliers, n'ayant pas d'impacts directs sur la sécurité ou liés aux revendications (*claims*) ou à l'authentification composée (*compound authentication*) ;
- ***ForestSpecific*** : SID considéré comme propre à une forêt. Il s'agit de SID de domaine (c'est-à-dire de la forme `S-1-5-21-X-Y-Z-RID`), où RID<500, RID=519 (Administrateurs de l'entreprise) ou RID=520 (Administrateur du schéma) ;
- ***DomainSpecific*** : SID considéré comme propre à un domaine. Il s'agit de SID de domaine (c'est-à-dire de la forme `S-1-5-21-X-Y-Z-RID`), où $500 \leq$ RID $<1\,000$. Cette notion de SID a disparu depuis Windows Server 2012 et est désormais considérée comme de type *ForestSpecific* ;
- ***SID en quarantaine*** : le SID n'appartient pas au domaine approuvé.

La politique de filtrage définit alors les SID qui sont conservés et ceux qui sont filtrés suivant la nature de la relation d'approbation :

- ***WithinForest*** :
  - les SID de type *AlwaysFilter* sont filtrés,
  - les SID de type *DomainSpecific* et *AlwaysFilter* sont filtrés ;
- ***QuarantinedWithinForest*** :
  - les SID de type *AlwaysFilter* sont filtrés,
  - les SID de type *DomainSpecific* et *ForestSpecific* sont filtrés,
  - les SID en quarantaine sont filtrés ;
- ***CrossForest*** :
  - les SID de type *AlwaysFilter* sont filtrés,
  - les SID de type *DomainSpecific* et *ForestSpecific* sont filtrés,
  - les SID doivent appartenir à la forêt approuvée (c'est-à-dire au moins à un domaine) ;
- ***External*** :
  - les SID de type *AlwaysFilter* sont filtrés,
  - les SID de type *DomainSpecific* et *ForestSpecific* sont filtrés ;
- ***QuarantinedExternal*** :
  - les SID de type *AlwaysFilter* sont filtrés,
  - les SID de type *DomainSpecific* et *ForestSpecific* sont filtrés,
  - les SID en quarantaine sont filtrés.

Note : un traitement particulier est effectué pour le SID `ENTERPRISE_DOMAIN_CONTROLLERS` (S-1-5-9), c'est-à-dire le SID affecté à tous les contrôleurs de domaine d'une forêt. Ce SID est conservé

uniquement pour les relations de type intraforêt.

6.6 Sécurité des relations

Dans la pratique, toutes les relations entre domaines d'une même forêt sont de type *WithinForest* et il est très rare (et déconseillé [6]), de positionner la propriété `QUARANTINED_DOMAIN` sur ce type de relation. Ainsi, la politique de filtrage *WithinForest* s'applique et tous les contrôleurs de domaine de la forêt ne filtrent que les SID catégorisés comme *AlwaysFilter* ou *DomainSpecific*.

Si un attaquant arrive à compromettre un domaine, il est alors en mesure d'utiliser le mécanisme des *referral tickets* pour usurper une identité privilégiée sur un domaine approuvé. De fil en fil, il peut compromettre tous les domaines de la forêt. La compromission d'un domaine entraîne donc la compromission de tous les domaines de la forêt. D'où la phrase dogmatique : « la frontière d'administration est le domaine, la frontière de sécurité est la forêt ».

Pour les relations extraforêt, la sécurité dépend des paramètres de la relation. Il faut cependant être vigilant avec les relations ayant une politique de filtrage de type *External* : le filtrage mis en œuvre dans ce cas est insuffisant et permet de réaliser des élévations de privilèges en usurpant des identités d'un domaine à l'autre.

Dans ce cas, il est recommandé d'activer le filtrage des SID [18] afin de filtrer tous les SID n'appartenant pas au domaine distant.

7 Délégation

7.1 Principe de la délégation

Comme abordé dans la partie 2.5, après authentification d'un utilisateur, un serveur peut utiliser le mécanisme de l'impersonation afin de prendre l'identité d'un utilisateur authentifié pour effectuer le contrôle d'accès à ses ressources locales. Cependant, sans la délégation d'authentification, le serveur serait incapable de s'authentifier à distance au nom de l'utilisateur dans un contexte d'*impersonation*.

18. Via la commande `netdom /quarantine`.

La délégation d'authentification consiste donc à permettre à un serveur, après authentification d'un utilisateur, de s'authentifier au nom de celui-ci auprès d'autres serveurs. Pour ce faire, pendant ou après l'authentification d'un client, le serveur doit récupérer des éléments nécessaires pour réaliser les authentifications distantes. Tous les protocoles d'authentification ne se prêtent pas à la délégation, en particulier ceux basés sur des défi-réponses (NTLM, Digest, etc.). En effet, avec ce type de protocole, les empreintes des utilisateurs devraient être transmises au serveur, ce qui n'est pas acceptable. En revanche, Kerberos, de par son mécanisme des tickets, est particulièrement adapté à la délégation : si celle-ci est autorisée, un ticket et une clé de session associée sont transmis au serveur et permettent à celui-ci de s'authentifier au nom du client. Ces éléments, ticket et clé, pourront soit être transmis par le client lui-même, soit récupérés par le serveur auprès d'un KDC.

La délégation étant une opération sensible du point de vue de la sécurité, elle est contrôlée par plusieurs paramètres décrits ci-dessous. Dans tous les cas, il est possible de désactiver la capacité de délégation d'un compte en positionnant l'option `ADS_UF_NOT_DELEGATED` sur l'attribut `userAccountControl` [23] du compte dans l'annuaire. Par défaut, dans un Active Directory, tous les comptes peuvent être délégués (l'option `ADS_UF_NOT_DELEGATED` n'est positionnée sur aucun compte). La restriction de délégation des comptes sensibles, en particulier ceux d'administration, est une mesure de sécurisation fortement recommandée.

7.2 Délégation complète

La délégation complète (*unconstrained delegation*[19]) est apparue dès Windows 2000 et reprend celle proposée dans la RFC 1510. Celle-ci permet à un client de déléguer complètement son authentification à un serveur en lui transmettant un TGT ainsi que la clé de session associée. Avec ces deux éléments, le serveur est alors en mesure de demander à un KDC des tickets de service au nom de l'utilisateur pour n'importe quel service.

Dans l'Active Directory, l'activation de cette forme de délégation requiert deux conditions :

19. D'autres termes peuvent exister tels qu'*open delegation*, *full delegation*, *general delegation*, etc.

- le serveur doit préalablement être autorisé pour la délégation complète en positionnant l'option `ADS_UF_TRUSTED_FOR_DELEGATION` sur l'attribut `userAccountControl` [23] du compte du serveur ;
- le client ne doit pas être interdit de délégation (cf. paragraphe précédent).

Lorsqu'un client est autorisé pour la délégation complète, l'attribut `forwardable` est activé dans les options de son TGT (figure 5, étape ①). De même, un client est informé qu'il se connecte à un serveur approuvé pour la délégation via l'attribut `ok-as-delegate` activé dans les options du ticket de service reçu par le client (figure 5, étape ②). Si les deux conditions sont réunies, le SSP Kerberos décide, automatiquement et systématiquement, de déléguer l'authentification.

Pour réaliser cette délégation, le client doit envoyer au serveur un TGT à son nom. Il ne peut cependant pas envoyer son TGT initial, car celui-ci est potentiellement restreint à son adresse réseau [20]. Un second TGT est donc demandé [21] [22] (figure 5, étape ③) : il s'agit d'un TGT ayant l'attribut `forwarded` positionné, qui stipule que le ticket n'est plus restreint à l'adresse du client (le terme de *forwarded TGT* sera utilisé dans la suite pour désigner ce TGT). Un TGT avec l'option `forwarded` n'est délivré que si le TGT servant à authentifier la demande possède l'attribut `forwardable`.

Une fois en possession du *forwarded TGT*, le client doit l'envoyer avec la clé de session associée au serveur. Le processus de transfert est décrit dans la RFC 4121 : dans le message `KRB_AP_REQ` d'authentification du client auprès du serveur, le *checksum* du champ `cksum` est modifié pour inclure une structure de type `KRB_CRED`. Celle-ci contient alors le ticket et la clé de session (respectivement dans les champs `tickets` et `enc-part.ticket-info.key`) (figure 5, étape ④).

20. Même si, comme vu dans la partie 1.9, ceci n'est pas activé par défaut dans les environnements Active Directory. Dans tous les cas, l'adresse du client étant dans la partie chiffrée du ticket, le client ne peut pas savoir si la restriction est effective ou non et un second TGT est systématiquement demandé.

21. Ce TGT est demandé via le service TGS et pas le service AS. Il est donc nécessaire de disposer au préalable d'un TGT valide. N'ayant pas été obtenu par le service AS, l'attribut `initial` ne sera pas positionné dans les options de ce nouveau TGT.

22. Cela explique pourquoi deux TGT sont généralement présents pour une session d'authentification donnée (visibles avec la commande `klist`).

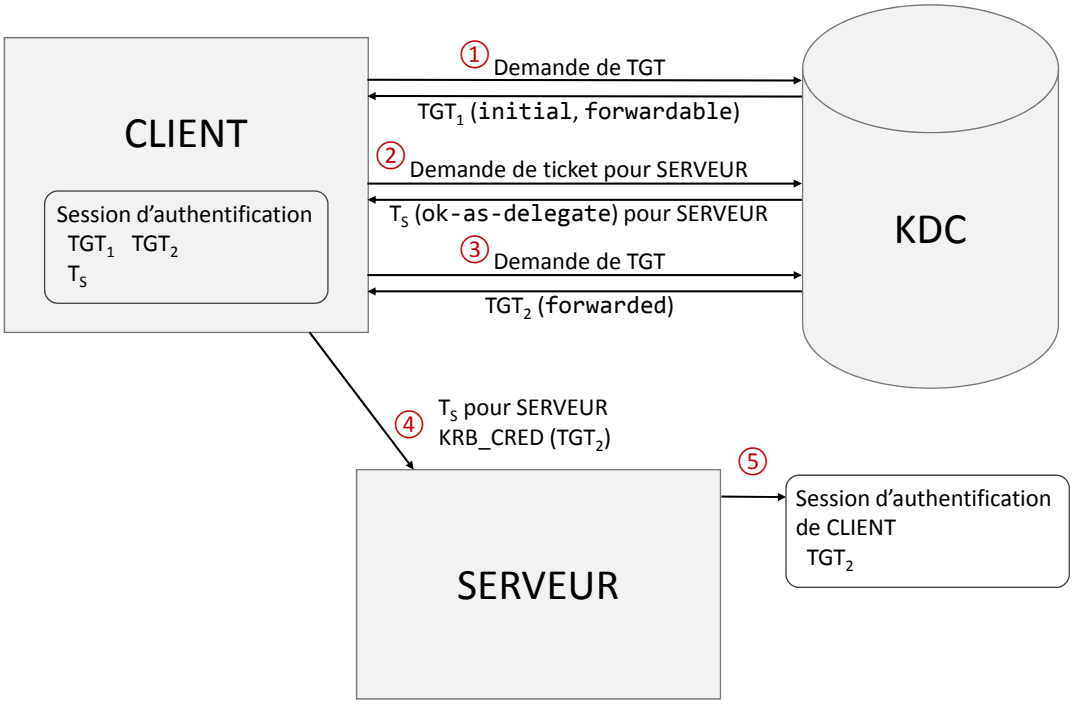

Figure 5. Synopsis de la délégation complète

En recevant un message `KRB_AP_REQ`, le serveur regarde si un *forwarded TGT* est présent dans la requête. Dans l'affirmative, le *forwarded TGT* est récupéré et conservé dans les données de la session d'authentification associée à l'utilisateur (figure 5, étape ⑤). La présence du TGT permet de donner une capacité de délégation : en cas d'authentification distante sur un serveur tiers, l'opération est réalisée non pas dans le contexte d'authentification du serveur, mais dans celui du client.

7.3 Risques associés à la délégation complète

Au vu de son mode de fonctionnement, la délégation complète est un mécanisme risqué. Un client qui transmet à un serveur son TGT et la clé de session associée n'a alors plus aucun contrôle dessus. Avec ces éléments, le serveur a la possibilité de demander des tickets de service au nom du client pour n'importe quel autre serveur.

Si un attaquant prend le contrôle d'un serveur approuvé pour la délégation, il est en mesure d'énumérer les sessions d'authentification en cours sur la machine et de récupérer les TGT et clés associées de tous les utilisateurs authentifiés ou qui se sont récemment authentifiés auprès de

ce serveur. La réutilisation de ces TGT est d'autant plus facile qu'ils ne sont pas restreints à une adresse réseau donnée.

Dans une configuration par défaut d'un Active Directory, seuls les contrôleurs de domaine sont, structurellement, approuvés pour la délégation complète. L'approbation d'un autre serveur est une opération ayant des conséquences de sécurité importantes et ne doit être effectuée qu'en cas de nécessité. Dans tous les cas, tout serveur approuvé pour la délégation doit être considéré comme aussi sensible qu'un contrôleur de domaine.

8 Service for User (S4U)

Afin d'apporter plus de finesse, et donc de sécurité, dans la délégation, Windows 2003 a vu l'apparition de deux nouvelles extensions venues compléter le protocole Kerberos. Regroupées sous la norme *Service for User* [10] (S4U), ces extensions, *Service-for-User-to-Self* (*S4U2self*) et *Service-for-User-to-Proxy* (*S4U2proxy*), apportent les fonctionnalités suivantes :

- la transition de protocole (*protocol transition*) ;
- la délégation contrainte (*constrained delegation*).

8.1 Service-for-User-to-Self (S4U2self)

S4U2self permet à un serveur d'obtenir un ticket de service pour lui-même tout en spécifiant arbitrairement le nom du client du ticket. Cette opération est effectuée via un message `KRB_TGS_REQ` en ajoutant, dans la partie *preauthentication data* (`padata`), un bloc de type `PA-FOR-USER` qui indique au KDC le nom du client désiré pour le ticket (qui, dans une requête traditionnelle, devrait être le nom du demandeur, c'est-à-dire, dans ce cas, le nom du serveur qui effectue la demande). Via l'API, cette action est réalisée via un appel à la fonction `LsaLogonUser` avec un message de type `KerbS4ULogon`. Un exemple de code est donné sur [2].

Le serveur, destinataire du ticket, va utiliser la PAC (cf. partie 2.5) incluse dans le ticket S4U2self pour constituer un jeton de sécurité (*access token*) représentant le contexte de sécurité du client spécifié. Une application de l'extension S4U2self permet donc à un serveur de disposer d'un jeton de sécurité pour n'importe quel utilisateur. Ce jeton peut ensuite être utilisé pour :

– l'identification du client (niveau `SecurityIdentification`) en utilisant les fonctions traditionnelles de l'API afin de déterminer les groupes d'appartenance présents dans le jeton ;
– le contrôle d'accès (niveau `SecurityImpersonation`) à ses ressources locales en affectant le jeton à un *thread* afin d'emprunter l'identité du client (mécanisme d'*impersonation*) pour le contrôle d'accès.

Le niveau d'*impersonation* du jeton [21] est, par défaut, `SecurityIdentification`. Cependant, si le serveur dispose du privilège `SeTcbPrivilege` [23], le niveau est `SecurityImpersonation`.

Une description détaillée de S4U2self et la présentation d'autres utilisations peuvent être trouvées dans [26].

Point important pour la partie ci-dessous : si, dans l'Active Directory, l'option `ADS_UF_TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION` (`T2A4D`) est activée dans le champ `userAccountControl` du compte machine associé au serveur, le ticket S4U2self aura l'option `forwardable` activée.

8.2 Service-for-User-to-Proxy (S4U2proxy)

S4U2proxy permet à un serveur d'obtenir un ticket de service pour un autre service tout en spécifiant arbitrairement le nom du client du ticket. Cette opération est effectuée via un message `KRB_TGS_REQ` ayant les propriétés suivantes :

– l'attribut `cname-in-addl-tkt` est activé dans les options de ticket demandées (champ `TGS-REQ.req-body.kdc-options`). Ceci indique au service TGS que le nom du client dans le ticket devant être créé ne doit pas être celui qui effectue la demande (c'est-à-dire le serveur), mais celui spécifié par le ticket inclus dans la partie `KDC-REQ-BODY` (champ `TGS-REQ.req-body.additional-tickets`) ;
– le champ `TGS-REQ.req-body.additional-tickets` contient un ticket (ce champ est généralement absent dans une requête classique).

Le ticket inclus par le serveur dans le champ `TGS-REQ.req-body.additional-tickets` peut être obtenu de deux manières différentes. Si le client s'est authentifié par Kerberos, il s'agit du ticket de service présenté le client lorsqu'il s'est authentifié auprès du serveur. Ce ticket doit cependant avoir l'attribut `forwardable` activé dans les options du ticket. Cet attribut n'est positionné par le KDC qu'à

23. Appelé également « Agir en tant que partie du système d'exploitation ».

condition que le compte du client soit autorisé à être délégué (cf. partie 7.2).

Si le client ne s'est pas authentifié par Kerberos, le serveur peut utiliser le mécanisme S4U2self présenté ci-dessus pour demander un ticket attribué au client pour le serveur. Là encore, ce ticket ne peut être présenté dans une requête S4U2proxy que si l'attribut `forwardable` est activé (ce qui est le cas si `ADS_UF_TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION` est positionné dans le champ `userAccountControl` du compte du serveur). Ce scénario est utilisé lorsque le client ne s'authentifie pas en Kerberos auprès du serveur alors que le serveur doit s'authentifier en Kerberos au nom du client auprès d'un autre service. Il constitue donc le principe de la transition de protocole (*protocol transition*).

Cependant, le serveur ne peut obtenir de ticket pour n'importe quel service. Le SPN du service demandé doit préalablement être autorisé en étant présent dans l'attribut `msDS-AllowedToDelegateTo` (`A2D2`) du compte de machine associé au serveur. Ainsi, contrairement à la délégation complète qui permet au serveur d'obtenir un ticket pour n'importe quel service, S4U2proxy impose que les services pour lesquels un ticket est demandé soient préalablement déclarés dans l'Active Directory. Cette exigeance constitue le principe de la délégation contrainte (*constrained delegation*). Une seconde restriction, introduite avec Windows Server 2012, permet de spécifier, via l'attribut `msDS-AllowedToActOnBehalfOfOtherIdentity` [10] les entités de sécurité autorisées à se connecter à un service donné.

Il faut noter que, lorsqu'un serveur emprunte l'identité d'un utilisateur via le mécanisme de l'*impersonation* et qu'il doit s'authentifier à distance au nom de cet utilisateur, le SSP Kerberos commence par tenter la délégation complète en regardant si un TGT est présent dans la session d'authentification de l'utilisateur. Si un *forwarded TGT* est présent, il est utilisé pour demander un ticket de service au nom du client. Si aucun TGT n'est présent, la délégation contrainte est tentée via une requête S4U2proxy. Cette extension est donc bien plus souvent utilisée qu'il n'y paraît.

8.3 Sécurité des extensions S4U

S4U2self ne présente pas de risque particulier. Il est certes possible, par cette extension, d'obtenir un jeton de sécurité ayant un niveau

`SecurityImpersonation` pour n'importe quel utilisateur du domaine (autorisé pour la délégation), mais cela nécessite le privilège `SeTcbPrivilege`. Or, par défaut, ce privilège n'est accordé qu'à l'entité `LocalSystem` et il est fortement déconseillé de le concéder à d'autres utilisateurs.

Quant à S4U2proxy, cette extension permet la délégation contrainte qui améliore grandement la sécurité de la délégation en permettant de restreindre les services destinataires. En revanche, si la transition de protocole est activée pour un serveur, celui-ci a la possibilité de s'authentifier auprès de tous les services autorisés sous l'identité de n'importe quel utilisateur du domaine (autorisé à être délégué). Il convient donc de considérer ce type de serveur tout aussi sensible qu'un contrôleur de domaine, au même titre que ceux autorisés pour la délégation complète.

9 Problématique du compte krbtgt

Comme décrit dans la partie 2.2, la clé K_{KDC} [24] est stockée au sein de l'Active Directory dans les attributs du compte `krbtgt`. Or cette clé doit être commune à tous les contrôleurs de domaine afin qu'un TGT ou qu'une signature PAC générés par un contrôleur de domaine donné soient valides auprès d'un autre contrôleur.

La synchronisation de cette clé est assurée par le mécanisme de réplication de l'Active Directory [7] qui se charge de recopier les attributs des objets de l'annuaire Active Directory sur tous les contrôleurs de domaine.

Ainsi, après un changement du mot de passe du compte `krbtgt`, dans un laps de temps plus ou moins long suivant la complexité de l'annuaire et du réseau (mais tout de même de l'ordre de quelques minutes), les nouvelles clés sont transmises à tous les contrôleurs de domaine via la réplication de l'Active Directory.

Après un changement de clé K_{KDC}, tous les TGT et les PAC signés devraient se voir invalidés. Ce cas est toutefois prévu : si un client présente un TGT chiffré avec une mauvaise clé K_{KDC}, le KDC lui retourne l'erreur `KRB_AP_ERR_MODIFIED` indiquant au client que son TGT est invalide.

24. En réalité, *les* clés K_{KDC} puisque plusieurs types de clés sont présents (NTLM, AES, DES).

Celui-ci doit effacer le TGT de son cache, puis en redemander un nouveau. Dans pareil cas, le changement des clés K_{KDC} devrait provoquer un pic de renouvellement des TGT.

Cependant, après un changement de clé K_{KDC}, un problème plus complexe surviendrait : les contrôleurs qui posséderaient l'ancienne clé K_{KDC} ne seraient plus en état de récupérer la nouvelle clé. En effet, ne possédant que l'ancienne clé K_{KDC}, les TGT chiffrés avec cette clé ne pourraient pas être déchiffrés (et donc acceptés) par les contrôleurs disposant de la nouvelle clé. Les contrôleurs de domaine avec l'ancienne clé seraient donc dans l'incapacité de s'authentifier auprès des contrôleurs disposant de la nouvelle et seraient donc incapables de récupérer la nouvelle clé K_{KDC}.

Pour éviter ces problèmes, Microsoft a choisi une solution simple et efficace : pour le service TGS, les clés valables sont celles correspondant au mot de passe courant et au mot de passe précédent. Lors du traitement d'un TGT, un contrôleur de domaine commence par essayer de le déchiffrer avec la clé courante et, en cas d'échec, réitère l'opération avec la clé précédente [25]. C'est seulement en cas de second échec que l'erreur `KRB_AP_ERR_MODIFIED` est renvoyée au client.

Dans les faits, le changement du mot de passe du compte `krbtgt` n'est effectué automatiquement que lors du passage du niveau de fonctionnalité d'un domaine de 2003 (ou inférieur) vers 2008 (ou supérieur). Ce changement a pour objectif de générer les clés AES associées au compte et apparues avec Windows Server 2008. Il est également possible de changer manuellement les clés en procédant à la réinitialisation du mot de passe du compte `krbtgt`. Mais, si l'on souhaite changer complètement les clés K_{KDC}, il faut procéder à deux changements successifs du mot de passe.

Cette procédure de double changement est plus complexe qu'il n'y paraît. En effet, si plusieurs contrôleurs de domaine sont présents, il est nécessaire de laisser suffisamment de temps, après le premier changement de mot de passe, pour s'assurer que la réplication s'est terminée et que tous les contrôleurs ont reçu la nouvelle clé K_{KDC}. Passée cette première étape, il est possible de procéder au second changement. Cependant, le laps

25. Comme vu dans la partie 2.3, l'Active Directory conserve toujours les clés correspondant au mot de passe courant ainsi que celles correspondant au mot de passe précédent.

de temps entre ces deux changements offre une fenêtre pendant laquelle un attaquant pourrait générer un TGT (chiffré avec l'ancienne clé K_{KDC}) et s'octroyer des droits lui permettant de récupérer la nouvelle clé. Le changement du mot de passe du compte `krbtgt` nécessite donc d'isoler complètement les contrôleurs pendant les deux changements. Dans la pratique, cette opération est extrêmement compliquée (cf. partie 11.5).

10 Journalisation

La journalisation est généralement un moyen assez efficace pour déceler une anomalie ou des comportements suspects. Deux types d'évènements sont importants à surveiller :

- les évènements liés spécifiquement à Kerberos, majoritairement générés sur les contrôleurs de domaine ;
- les évènements liés à l'authentification d'un utilisateur, générés sur tous les systèmes prenant part à l'authentification.

Note : les identifiants d'évènements décrits ci-dessous le sont par rapport au fournisseur de sécurité des systèmes Vista et supérieurs (`Microsoft-Windows-Security-Auditing`). Pour les systèmes Windows XP ou 2003, les évènements présentés sont, dans leur grande majorité, également générés par le fournisseur `Security`. La correspondance des identifiants d'évènements entre les deux fournisseurs est effectuée en retranchant 4096.

10.1 Évènements Kerberos

La première catégorie d'évènements intéressants liés à Kerberos concerne l'émission, par les KDC, de tickets aux utilisateurs. Ces évènements sont répartis en deux sous-catégories : les demandes de TGT sur le service d'authentification AS (sous-catégorie *Kerberos Authentication Service*) et les demandes de ticket sur le service de délivrance de ticket TGS (sous-catégorie *Kerberos Service Ticket Operations*). Le tableau 2 recense ces évènements.

Par exemple, lorsqu'un TGT est délivré par un contrôleur de domaine, un événement 4768 est généré dont la description est la suivante :

Table 2. Évènements Kerberos

EventID	Service	Description
4768	AS	A Kerberos authentication ticket (TGT) was requested
4771	AS	Kerberos pre-authentication failed
4772	AS	A Kerberos authentication ticket request failed
4769	TGS	A Kerberos service ticket was requested
4770	TGS	A Kerberos service ticket was renewed
4773	TGS	A Kerberos service ticket request failed

```
Un ticket d'authentification Kerberos (TGT) a été demandé.

Informations sur le compte :
   Nom du compte :                    %1 (ex : Administrateur)
   Nom du domaine Kerberos fourni : %2 (ex : DOMAINE)
   ID de l'utilisateur :              %3 (ex : DOMAINE\Administrateur)

Informations sur le service :
   Nom du service :           %4 (ex : krbtgt)
   ID du service :            %5 (ex : DOMAINE\krbtgt)

Informations sur le réseau :
   Adresse du client :        %10 (adresse IPv4 ou IPv6 du client)
   Port client :              %11 (port source de la requête)

Informations supplémentaires :
   Options du ticket :              %6 (ex : 0x40810010)
   Code de résultat :               %7 (0 : requête aboutie / TGT émis)
   Type de chiffrement du ticket : %8 (ex : 0x12 pour aes256-cts-hmac-sha1-96)
   Type de préauthentification :  %9 (2: pa-enc-timestamp, 15/17: PA_PK_AS_REP)

Informations sur le certificat :
   Nom de l'émetteur du certificat :   %12 (CN de l'autorité)
   Numéro de série du certificat :     %13
   Empreinte numérique du certificat : %14
```

Les informations sur le certificat sont fournies uniquement si un certificat a été utilisé pour la préauthentification (extension PKINIT décrite dans la partie 3). Les types de préauthentification, les options de ticket, les types de chiffrement et les codes de résultat sont définis dans la RFC 4120.

Il est important de noter que le seul élément journalisé permettant de connaître la machine depuis laquelle la requête de TGT a été effectuée est son adresse IP, et non le nom NetBios ou DNS de la machine, par exemple.

De même, dans le cas de la délivrance d'un ticket de service, un événement 4769 est généré.

```
Un ticket de service Kerberos a été demandé.
Informations sur le compte :
   Nom du compte :                %1  (ex : Administrateur@DEMO.TEST)
   Domaine du compte :            %2  (ex : DEMO.TEST)
   GUID d'ouverture de session : %10 (ex : 5fc6adab-2c07-40d4-c7ce-c79287c3a58f)

Informations sur le service :
   Nom du service :   %3 (ex : DC-2012-01$)
   ID du service :    %4 (ex : DEMOTEST\DC-2012-01$)

Informations sur le réseau :
   Adresse du client :  %7 (adresse IPv4 ou IPv6 du client)
   Port client :               %8 (port source de la requête)
Informations supplémentaires :
   Options du ticket :              %5
   Type de chiffrement du ticket : %6
   Code d'échec :                   %9
   Services en transit :           %11
```

Là encore, seule l'adresse IP est journalisée. Il faut également souligner que, concernant le nom du service demandé pour le ticket, seul le nom du serveur est journalisé et non le SPN complet (par exemple `serveur-01` au lieu de `cifs/serveur-01.demo.test`). Enfin, que ce soit pour l'émission des TGT ou des tickets de service, les données contenues dans une éventuelle PAC ne sont jamais journalisées.

Ainsi, en cas de suspicion d'utilisation, par un attaquant, d'un compte compromis, l'analyse des journaux des contrôleurs de domaine [26] permet, via les évènements liés à l'émission de ticket, de connaître :

- l'adresse IP de la machine depuis laquelle l'attaquant opère. En cas d'utilisation d'adresses dynamiques, le travail d'analyse est alors plus complexe et nécessite d'autres journaux, en particulier ceux des serveurs DHCP ;
- les machines pour lesquelles l'attaquant a demandé un ticket de service, sans toutefois avoir d'information précise sur le service demandé.

Il faut souligner que ces évènements indiquent uniquement à quel moment des tickets ont été émis à un utilisateur donné. Il n'est donc pas possible, avec uniquement ce type d'évènements, de savoir :

26. Il est important de préciser que les journaux de tous les contrôleurs de domaine sont nécessaires. En effet, un ticket (TGT ou ticket de service) peut théoriquement être demandé à n'importe quel contrôleur.

– si un utilisateur s'est effectivement authentifié sur une machine (un ticket peut être demandé sans qu'il soit utilisé par la suite) ;
– à quel moment précis l'utilisateur s'est authentifié (un ticket peut être demandé et être utilisé plusieurs heures après) ;
– combien de fois l'utilisateur s'est authentifié (un ticket peut servir à s'authentifier pendant toute sa durée de validité).

D'autres évènements, récapitulés dans le tableau 3, sont également importants à surveiller.

Table 3. Évènements de sécurité

EventID	Sous-catégorie	Description
4675	Audit Logon	SIDs were filtered
4649	Audit Other Logon/Logoff Events	A replay attack was detected
5378	Audit Other Logon/Logoff Events	The requested credentials delegation was disallowed by policy

L'évènement 4675 est généré lorsqu'un contrôleur de domaine traite un *referral ticket* et qu'un ou plusieurs SID sont filtrés conformément à la politique de filtrage (cf. partie 6.5) liée à la relation d'approbation entre le domaine du contrôleur où l'évènement est généré et un domaine tiers. Normalement, aucun SID ne devant être filtré, la présence de cet évènement indique soit :

– une mauvaise configuration du domaine distant (où des SID sont encore présents dans les champs *sIDHistory* des comptes utilisateurs) ;
– la compromission du domaine distant et la tentative, par un attaquant, de rebond sur un domaine approuvé.

Dans tous les cas, il est indispensable d'analyser la liste des SID filtrés indiqués par l'évènement.

L'évènement 4649 est généré lorsqu'un rejeu [27] d'authentifiant Kerberos est détecté lors du traitement d'un message `KRB_AP_REQ`. Il peut donc être émis par tout système prenant part à l'authentification Kerberos. Là encore, cet évènement peut indiquer une erreur de configuration (problème de date), un problème réseau (surtout en UDP) ou une tentative d'attaque

27. Pour rappel de la partie 1.9, un rejeu est détecté lorsqu'un authentifiant reçu n'est plus valide (pour les services AS, TGS et AP) ou qu'il a déjà été reçu (service AP).

par rejeu d'authentifiant.

Enfin, il existe également d'autres fournisseurs de journalisation (*event log provider*), plus ou moins liés à Kerberos :

- Microsoft-Windows-KdsSvc ;
- Microsoft-Windows-Kerberos-KdcProxy ;
- Microsoft-Windows-Kerberos-Key-Distribution-Center ;
- Microsoft-Windows-Security-Kerberos.

Malheureusement, la liste des évènements [28] pouvant être émis par ces fournisseurs montre que ceux-ci ne sont utiles qu'à des fins de diagnostic ou de résolution de problèmes et sont peu utiles du point de vue de la sécurité.

10.2 Événements d'authentification

La seconde famille d'évènements intéressants à étudier concerne les évènements liés à l'authentification, qu'elle soit réalisée par Kerberos ou par tout autre protocole d'authentification. Les principaux évènements sont inventoriés dans le tableau 4.

Table 4. Évènements d'authentification

EventID	Sous-catégorie	Description
4624	Audit Logon	An account was successfully logged on
4625	Audit Logon	An account failed to log on
4634	Audit Logoff	An account was logged off
4647	Audit Logoff	User initiated logoff
4964	Audit Special Logon	Special groups have been assigned to a new logon
4672	Audit Sensitive Privilege Use	Special privileges assigned to new logon

Ainsi, sur chaque système où un utilisateur tente une authentification, un évènement est généré : 4624 en cas d'authentification réussie, 4625 en cas d'échec.

L'évènement 4624 est particulièrement important car il permet de tracer précisément les authentifications effectives sur chaque système de

28. La liste des évènements peut être obtenue par la commande `wevtutil gp nom-fournisseur /ge:true /gm:true`

tous les utilisateurs, qu'ils soient membres d'un domaine ou local à un système. Sa description (partielle) est la suivante :

```
L'ouverture de session d'un compte s'est correctement déroulée.

Sujet :
   ID de sécurité :               %1
   Nom du compte :                %2
   Domaine du compte :            %3
   ID d'ouverture de session :    %4

Type d'ouverture de session :     %9
   (2:interactif, 3:network, 6:service, 10:remoteinteractive, etc.).

Nouvelle ouverture de session :
   ID de sécurité :               %5
   Nom du compte :                %6
   Domaine du compte :            %7
   ID d'ouverture de session :    %8
   GUID d'ouverture de session :  %13

Informations sur le processus :
   ID du processus :      %17
   Nom du processus :     %18

Informations sur le réseau :
   Nom de la station de travail :  %12
   Adresse du réseau source :      %19
   Port source :                   %20

Informations détaillées sur l'authentification :
   Processus d'ouverture de session :   %10
   Package d'authentification :         %11
   Services en transit :                %14
   Nom du package (NTLM uniquement) :   %15
   Longueur de la clé :                 %16
```

Les différents champs sont plus ou moins remplis suivant le type d'ouverture de session et le protocole utilisé. Cependant, lorsque Kerberos est mis en œuvre et dans le cas d'une authentification réseau (type `LOGON32_LOGON_NETWORK`), là encore, seule l'adresse IP de la machine distante est journalisée (champ %19).

Toujours dans cas de suspicion d'utilisation, par un attaquant, d'un compte compromis, les évènements d'authentification sont bien plus utiles et précis que les évènements liés à l'émission de tickets. La contrepartie est qu'il faut récupérer ces évènements sur tous les systèmes. Ceci est rapidement fastidieux, mais nécessaire, *a minima* sur les systèmes sensibles (contrôleurs de domaine, serveurs et postes d'administration).

L'évènement 4624 permet uniquement de connaître le nom de l'utilisateur authentifié, mais il ne mentionne pas les SID positionnés dans le jeton créé. Il n'est donc pas possible, avec cet évènement, de déceler une éventuelle modification de la PAC ayant permis à un attaquant d'ajouter arbitrairement des SID à l'utilisateur.

L'évènement 4672 peut permettre de détecter une éventuelle modification de PAC. En effet, lors de la création d'une session d'authentification, si la politique de sécurité d'un système confère aux SID du jeton des privilèges sensibles (*SeDebugPrivilege*, *SeBackupPrivilege*, etc. [4]), un évènement 4672 est généré. Celui-ci indique alors les privilèges sensibles accordés.

```
Privilèges spéciaux attribués à la nouvelle ouverture de session.

Sujet :
   ID de sécurité :            %1 (ex : DEMO\user)
   Nom du compte :             %2 (ex : user)
   Domaine du compte :         %3 (ex : DEMO)
   ID d'ouverture de session : %4 (identifiant de la session d'authentification)

Privilèges : %5 (liste des privilèges sensibles accordés :
                 SeSecurityPrivilege, SeLoadDriverPrivilege, etc.)
```

Dans une configuration standard, les privilèges sensibles ne sont attribués qu'aux membres des groupes d'administration ou d'opération ainsi qu'aux comptes de service. Ainsi, l'apparition d'un événement 4672 suite à l'authentification d'un utilisateur n'appartenant pas à un groupe accordant des privilèges sensibles, peut révéler une éventuelle modification de PAC dans un ticket Kerberos à des fins d'élévation de privilèges.

11 Protections

Afin de se prémunir contre certains types d'attaques, Microsoft ajoute, au fur et à mesure des versions successives de Windows, différents mécanismes de protection dont certains sont détaillés ci-dessous.

11.1 Protected users group

Le groupe *Protected users group* est un groupe utilisateur de l'Active Directory (RID de 516) apparu avec Windows Server 2012 R2 [18]. L'objectif de celui-ci est de renforcer la protection des secrets d'authentification

des utilisateurs membres de ce groupe. Ainsi, pour un membre de ce groupe, le comportement de certains composants d'authentification est modifié de la façon suivante :

- **côté contrôleur de domaine**, si le niveau de fonctionnalité du domaine est *a minima* Windows 2012 R2 :
  - les validations d'authentification via `NETLOGON` sont refusées, obligeant ainsi une authentification via Kerberos,
  - seuls des algorithmes de chiffrement basés sur AES sont acceptés,
  - toute forme de délégation est refusée,
  - la durée de vie des TGT est, par défaut, limitée à 4 heures ;
- **coté client**, pour les systèmes Windows prenant en compte ce groupe [29] :
  - les SSP msv1_0, WDigest et CredSSP ne mettent plus en cache des secrets d'authentification lorsqu'un utilisateur s'authentifie interactivement sur une machine. Il n'est alors plus possible de récupérer dans la mémoire de LSASS des secrets tels que le mot de passe ou l'empreinte NTLM de l'utilisateur,
  - pour Kerberos, uniquement les algorithmes de chiffrement basés sur AES sont supportés,
  - pour Kerberos, aucune clé dérivée du mot de passe n'est conservée en mémoire pour la session d'authentification de l'utilisateur (cf. partie 2.4).

L'objectif de ce groupe est donc de limiter au maximum la présence de secrets d'authentification dans la mémoire de LSASS. Il est donc conseillé d'y inclure le maximun d'utilisateurs sensibles, en particulier les comptes des administrateurs.

Cependant, comme toutes les protections, celle-ci n'est pas absolue. D'une part, localement à un système, il subsiste toujours dans la mémoire de LSASS les TGT et les clés de sessions associées des utilisateurs authentifiés. Cela laisse la possibilité à un attaquant de récupérer les TGT et de les utiliser à des fins illégitimes. D'autre part, pour récupérer ce TGT, l'attaquant doit disposer de privilèges d'administration sur le système, ce qui permet d'autres attaques (cf. conclusion de la partie 2.4).

Cependant, le dispositif du groupe *Protected users group* permet tout de même de limiter les secrets d'authentification récupérables. Il est plus

29. Pour l'instant, seuls les systèmes 6.3 prennent en compte ce groupe, mais Microsoft a annoncé le portage de la fonctionnalité sur les éditions précédentes de Windows.

contraignant de récupérer un TGT valable 4 heures, qu'une empreinte NTLM d'un mot de passe qui n'expire jamais. L'utilisation du dispositif du groupe *Protected users group* est donc recommandée.

11.2 Protected process

La notion de *protected process* est apparue avec Windows Vista, plus à des fins de DRM que de sécurité. Ce mécanisme permet d'interdire à tout processus en espace utilisateur l'ouverture d'un objet de type processus ou *thread* protégé (à l'exception de quelques droits utiles à des fins d'information). Initialement, seuls les processus liés à la protection des droits numériques (DRM) bénéficiaient de cette protection.

Avec Windows 8.1, Microsoft a étendu ce mode de fonctionnement et a activé le mécanisme sur les processus les plus critiques de Windows (smss.exe, csrss.exe, wininit.exe, winlogon.exe, services.exe). Une description plus approfondie peut être trouvée sur le blog d'Alex Ionescu [3].

Cependant, la protection n'est pas activée par défaut sur le processus LSASS, probablement lié au fait que de nombreux éditeurs chargent dans LSASS des extensions de type *SecurityProviders*, *Authentication Packages*, *Notification Packages*, *Security Packages*, etc. Or, le chargement de modules dans un *protected process* est fortement restreint et ceci aurait causé beaucoup de dysfonctionnements. Il est toutefois possible d'exécuter LSASS en tant que *protected process* en positionnant la clé `RunAsPPL`. Ainsi, tous les outils qui interagissent avec le processus LSASS dans le but de subtiliser des secrets d'authentification se verraient refuser l'ouverture du processus LSASS.

Évidemment, un attaquant ayant obtenu des droits d'administrateur sur un système peut contourner ce dispositif. Les méthodes actuelles passent généralement par le chargement d'un pilote dans le noyau [30] afin de désactiver l'attribut *protected process* dans la structure `_EPROCESS` ou en lisant les pages mémoire depuis le noyau.

Là encore, le dispositif des *protected process* n'est pas une protection absolue. Toutefois, il impose à un attaquant de réaliser des opérations supplémentaires, plus « bruyantes ».

30. Il faut noter qu'avec les éditions 64 bits de Windows, le chargement de pilote par le noyau requiert une signature numérique du pilote. Cela permet au moins de savoir, avec précision, quel outil a été utilisé.

11.3 Restricted admin

Lorsqu'un utilisateur s'authentifie auprès d'un serveur RDP, il lui transmet en clair [31] son mot de passe. Le processus d'authentification d'un utilisateur via RDP est identique à celui d'un utilisateur interactif : les SSP sont notifiés d'une authentification et récupèrent, à ce moment-là, le mot de passe afin de procéder à la mise en cache d'informations nécessaires pour permettre l'authentification implicite de l'utilisateur auprès de serveurs distants.

Ainsi, pour une session RDP d'un utilisateur, beaucoup de secrets d'authentification se trouvent dans la mémoire de LSASS. Ils peuvent être compromis en cas de prise de contrôle du système par un attaquant.

Le mode *Restricted admin* de RDP (du nom de l'option `/restrictedAdmin` du client Microsoft `mstsc.exe`) permet de changer la façon dont l'utilisateur doit s'authentifier. Le mot de passe n'est alors plus requis et il suffit de réussir une authentification distante. L'intérêt est de ne plus avoir aucun secret d'authentification associé à la session d'authentification (à condition que la délégation ne soit pas approuvée pour le serveur RDP). La contrepartie est qu'il devient possible de s'authentifier via une attaque de type *Pass-the-hash* ou *Pass-the-key*. De plus, l'administrateur doit être sensibilisé au fait qu'il n'est plus possible de s'authentifier auprès de serveurs tiers et qu'il ne doit jamais saisir son mot de passe dans la session RDP au risque d'annihiler tous les bénéfices de ce mode.

Le bénéfice/risque de ce mode doit donc être étudié avant d'être mis en œuvre. Cependant, couplé avec l'utilisation imposée d'une carte à puce pour l'authentification, ce mode offre des avantages avérés.

11.4 Abandon des anciens algorithmes

Dans les environnements Active Directory, trois générations de chiffrement sont supportées : DES, RC4 et AES (cf. partie 2.1). Ces algorithmes peuvent être utilisés pour :

– générer les parties chiffrées des messages des requêtes et des réponses entre un client, un KDC ou un serveur ;

31. Le terme « en clair » signifie ici que le serveur récupère en clair le mot de passe. Bien entendu, celui-ci est chiffré lors de son transfert sur le réseau. Ce transfert peut s'effectuer soit par déport des frappes clavier, soit par CredSSP.

– générer les parties chiffrées des tickets ;
– déterminer, pour le KDC, le type de clé de session à générer dans le ticket.

Le support des familles de chiffrement par les systèmes Windows est, dans une configuration par défaut, le suivant :

– Windows XP et Windows Server 2003 : DES et RC4
– Windows Vista et Windows Server 2008 : DES, RC4 et AES
– Windows 7 (et suivants) et Windows Server 2008 R2 (et suivants) : RC4 et AES. DES est supporté mais désactivé par défaut.

La sélection d'un algorithme de chiffrement dépend de plusieurs paramètres et de la version des systèmes qui les mettent en œuvre. Une description de son fonctionnement peut être trouvée en [27], [31] et [1].

Depuis Windows 7/2008R2, le paramètre « Sécurité réseau : Configurer les types de chiffrement autorisés pour Kerberos » [32] de la stratégie locale de sécurité permet de définir les algorithmes supportés par le SSP Kerberos pour le chiffrement des messages. Il permet essentiellement, si nécessaire, de réactiver DES sur ces systèmes.

Le choix de l'algorithme de chiffrement des tickets est différent et indépendant des capacités du client, celui-ci ne faisant que conserver les tickets sans avoir à les manipuler. Pour déterminer l'algorithme de chiffrement du ticket, le KDC commence par définir la liste des algorithmes supportés par le destinataire en analysant les attributs du compte (machine ou utilisateur) associée au service demandé pour le ticket :

– l'option `ADS_UF_USE_DES_KEY_ONLY` de l'attribut `userAccountControl` qui, si elle est activée, force l'utilisation d'algorithmes basés sur DES. Ceci est bien évidemment fortement déconseillé, sauf à des fins de compatibilité ;
– l'attribut `ms-DS-Supported-Encryption-Type` [13] (apparu depuis Windows Server 2008) qui permet d'indiquer les algorithmes supportés par le serveur. Cet attribut est en écriture pour le compte associé et doit normalement être mis à jour par celui-ci afin

32. Ce paramètre modifie la clé `HKLM\SOFTWARE\Microsoft\Windows\-CurrentVersion\Policies\System\Kerberos\Parameters\-SupportedEncryptionTypes`.

d'indiquer les algorithmes qu'il supporte [33].

Dans tous les cas, des algorithmes sont implicitement ajoutés (sauf si `ADS_UF_USE_DES_KEY_ONLY` est activé afin de forcer l'utilisation de DES) :

- AES pour compte `krbtgt` (pour les TGT) et les comptes des DC ou RODC (pour les tickets de service à destination de ces serveurs), si le niveau de fonctionnalité du domaine est, *a minima*, Windows Server 2008 ;
- RC4 et DES pour tous les comptes (y compris `krbtgt`).

Note : sur les systèmes Windows Server 2008 R2 et supérieurs, si DES est l'algorithme retenu mais qu'il n'a pas été réactivé, cela conduit à l'impossibilité d'utiliser Kerberos.

Une fois tous les algorithmes supportés par le destinataire du ticket identifiés, le KDC détermine celui à utiliser. Par défaut, la préférence est donnée aux algorithmes les plus sûrs (AES256, AES128, RC4 puis DES). Cependant, et uniquement pour les TGT et les *referral tickets*, si le paramètre `KdcUseRequestedEtypesForTickets` [27] est activé, la préférence est donnée suivant l'ordre de la liste des algorithmes proposée par le client dans le champ `KDC-REQ-BODY.etypes` de sa requête.

Le choix et le support des algorithmes de chiffrement des messages et des tickets dépend donc de divers paramètres et de la version des systèmes. Les recommandations suivantes peuvent être appliquées :

- vérifier qu'aucun compte n'a l'option `ADS_UF_USE_DES_KEY_ONLY` activée dans l'attribut `userAccountControl` ;
- s'assurer que l'attribut `ms-DS-Supported-Encryption-Type` de tous les comptes supportant des services Kerberos (compte de machine ou compte utilisateur utilisé comme compte de service Kerberos) est correctement positionné et que :
  - AES est activé pour tous les systèmes Windows Vista et Windows Server 2008 et supérieurs,
  - DES est désactivé pour tous les systèmes Windows 7 et Windows Server 2008 R2 et supérieurs ;
- s'assurer que tous les mots de passe ont été changés depuis une migration d'un serveur en contrôleur de domaine ou l'augmentation

33. Il est inutile de modifier ce paramètre : les systèmes Windows Vista et supérieurs mettent à jour, à chaque démarrage, cet attribut associé à leur compte machine afin d'indiquer leur support des algorithmes AES.

du niveau de fonctionnalité d'un domaine afin de s'assurer que les clés AES des utilisateurs sont générées dans l'annuaire Active Directory (cf. partie 2.3).

11.5 Comportement en cas de compromission

Lorsque le pire se produit, et qu'il est avéré qu'un attaquant a pris le contrôle des éléments les plus critiques d'un Active Directory, il est nécessaire de procéder à une opération de remédiation complexe et de grande envergure. Du fait des relations d'approbation, il est nécessaire dans un environnement multi-domaines de procéder à la remédiation simultanée de tous les domaines de la forêt.

Une remédiation consiste à reprendre le contrôle des éléments compromis. Une des mesures les plus emblématiques consiste à changer les mots de passe. Lors de l'inventaire des mots de passe à changer, deux types de compte doivent, en plus des comptes utilisateur, être pris en compte :

- le compte `krbtgt` dont la complexité et les risques associés au processus de changement du mot de passe (cf. partie 9) imposent l'isolation de tous les contrôleurs de domaine pendant l'opération ;
- les comptes de machine qui permettent d'élever les privilèges d'un attaquant sur chaque machine. Cette opération ne pouvant être réalisée sur l'intégralité du parc, elle concerne en général les systèmes les plus sensibles (contrôleur de domaine, poste d'administration, etc.) [34].

Microsoft propose désormais un guide d'explication et de préparation à ce type d'opération sous la terminologie « *Active Directory Forest Recovery* » [17,24].

11.6 Inventaire des recommandations

En guise de première conclusion, une liste de recommandations et de bonnes pratiques à adopter peut être établie.

Attribut UAC L'attribut userAccountControl permet de définir de nombreuses options relatives au compte associé. Parmi toutes les options décrites dans [23], il est nécessaire de s'assurer de la pertinence des options suivantes :

34. Cependant, la prudence impose de procéder à la réinstallation de ces machines pendant l'opération afin de s'assurer de l'absence de code malveillant.

- `ADS_UF_ENCRYPTED_TEXT_PASSWORD_ALLOWED` : indique que le mot de passe du compte est stocké en clair dans l'Active Directory. Ne devrait être activé sur aucun compte ;
- `ADS_UF_INTERDOMAIN_TRUST_ACCOUNT` : indique que le compte est un compte de *trust*. Doit être activé uniquement sur ce type de compte (cf. partie 6.2) ;
- `ADS_UF_SERVER_TRUST_ACCOUNT` : indique que le compte est un contrôleur de domaine, ce qui lui confère des droits supplémentaires. Doit être activé uniquement sur les comptes machine des contrôleurs de domaine ;
- `ADS_UF_DONT_EXPIRE_PASSWD` : indique que le mot de passe échappe aux politiques d'expiration du mot de passe. Ne devrait être activé sur aucun compte ;
- `ADS_UF_SMARTCARD_REQUIRED` : indique que l'utilisation des extensions PKINIT est requise (cf. partie 4.2) ;
- `ADS_UF_TRUSTED_FOR_DELEGATION` : indique que le compte associé est approuvé pour la délégation complète (cf. partie 7.2). Ne doit être activé que pour les comptes machine des contrôleurs de domaine et les comptes pour lesquels cela est absolument nécessaire ;
- `ADS_UF_NOT_DELEGATED` : indique que le compte ne peut être délégué (cf. partie 7.2). Devrait être positionné sur tous les comptes sensibles ;
- `ADS_UF_USE_DES_KEY_ONLY` : indique que le chiffrement DES est obligatoire pour les tickets (cf. partie 11.4). Ne doit être positionné sur aucun compte ;
- `ADS_UF_TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION` : indique que le compte est autorisé pour la transition de protocole (cf. partie 8.1). Ne doit être positionné que pour les comptes pour lesquels cela est absolument nécessaire ;
- `ADS_UF_DONT_REQUIRE_PREAUTH` : indique que la préauthentification Kerberos n'est pas requise (cf. partie 1.3). Ne devrait être positionné sur aucun compte.

Journalisation

- centraliser [35] (*a minima* pour les évènements inventoriés dans la partie 10) les journaux Windows d'un maximum de machines (serveurs sensibles, postes d'administration, etc.), et en particulier, les journaux de tous les contrôleurs de domaine ;

35. Le service de collecteur d'évènements, intégrés à Windows depuis Vista, et qui repose sur WinRM, peut être utilisé. Il a l'avantage de supporter la spécification d'une politique de remontée des journaux permettant, entre autres, de filtrer les évènements.

– mettre en place une politique d'analyse des journaux collectés. Cette politique doit notamment permettre la surveillance de l'utilisation des comptes privilégiés.

Relations d'approbation

– inventorier les relations d'approbation de tous les domaines d'une même forêt ;
– vérifier les propriétés des relations d'approbation (intraforêt et extraforêt) ;
– s'assurer que les relations de type *External* ont la propriété `QUARANTINED_DOMAIN` activée (cf. partie 6.5).

Délégation

– inventorier les comptes étant autorisés pour une des trois formes de délégation d'authentification et supprimer, si non nécessaire, la délégation ou les délégations sélectives :
  – délégation complète : option `ADS_UF_TRUSTED_FOR_DELEGATION` (cf. partie 11.6),
  – délégation sélective : champ `msDS-AllowedToDelegateTo` non vide,
  – délégation sélective avec transition de protocole : champ `msDS-AllowedToDelegateTo` non vide et option `ADS_UF_TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION` activée (cf. partie 11.6) ;
– interdire la délégation pour les comptes sensibles (option `ADS_UF_NOT_DELEGATED`, cf. partie 11.6).

Divers

– vérifier la SACL positionnée à la racine du domaine : celle-ci doit journaliser en particulier l'utilisation des *extended rights* (`ACTRL_DS_CONTROL_ACCESS`) pour tous les utilisateurs du domaine [36].

12 Conclusion

La compromission par un attaquant d'une base de comptes Active Directory se révèle être un évènement plus grave qu'on pourrait y penser.

36. En cas d'un accès à un objet de l'annuaire sur lequel un audit est activé, un évènement 4662 du fournisseur `Microsoft-Windows-Security-Auditing` est généré.

Outre la compromission des clés associées aux comptes utilisateur (qui peuvent être aisément changées), la compromission des clés du compte `krbtgt` et de celles des comptes machine a des impacts très importants.

En effet, avec les clés du compte `krbtgt`, un attaquant dispose des clés K_{KDC} qui lui permettent de forger des TGT et donc de pouvoir s'authentifier en usurpant l'identité de n'importe quel utilisateur du domaine. Cette situation est d'autant plus grave dans des environnements Active Directory, où Kerberos est également utilisé pour l'autorisation. Dans une telle situation, l'attaquant est alors en mesure de générer des TGT lui permettant de modifier l'appartenance aux groupes de sécurité à sa guise. Cette situation est aggravée par le fait que le changement des clés du compte `krbtgt` n'est pas trivial et que les attaques consistant à forger des PAC et des tickets ne sont pas évidentes à détecter. Enfin, dans le cas où les relations d'approbation sont mises en œuvre, les autres domaines sont également menacés.

Afin d'éviter ces scénarios catastrophes, la sécurité doit être intégrée à 3 niveaux :

- la **prévention** : la protection des bases Active Directory et des contrôleurs de domaine doit être une priorité absolue. Les dispositifs techniques de protection proposés par Microsoft doivent être utilisés dès que possible et les pratiques d'administration les plus rigoureuses possible ;
- la **supervision** : la centralisation des journaux de tous les systèmes doit être mise en place et leur analyse doit être capable de rapidement déceler toute anomalie ou incohérence ;
- la **réaction** : en cas de compromission d'une base de comptes Active Directory (de type *pwdump*), un plan approprié et global doit être mis en œuvre. Ce plan ne peut se réduire à un simple changement de quelques mots de passe.

Références

1. msds-supportedencryptiontypes - episode 1 - computer accounts. `http://samba.2283325.n4.nabble.com/attachment/2516039/0/Updated%20msDS-SupportedEncryptionTypes%20Episode%201.pdf`.
2. Keith Brown. Exploring s4u kerberos extensions in windows server 2003. `http://msdn.microsoft.com/fr-fr/magazine/cc188757.aspx`.
3. Alex Ionescu. The evolution of protected processes part 1 : Pass-the-hash mitigations in windows 8.1. `http://www.alex-ionescu.com/?p=97`.

4. Microsoft. Audit sensitive privilege use. http://technet.microsoft.com/en-us/library/dd772724.aspx.
5. Microsoft. Authentication protocol domain support (kerberos pac validation). http://msdn.microsoft.com/en-us/library/cc224020.aspx.
6. Microsoft. Configuring sid filter quarantining on external trusts. http://technet.microsoft.com/fr-fr/library/cc794757.aspx.
7. Microsoft. Directory replication service (drs) remote protocol. http://msdn.microsoft.com/en-us/library/cc228086.aspx.
8. Microsoft. Directory replication service remote protocol (issecretattribute). http://msdn.microsoft.com/en-us/library/dd207719.aspx.
9. Microsoft. Essential attributes of a trusted domain object. http://msdn.microsoft.com/en-us/library/cc223765.aspx.
10. Microsoft. Kerberos protocol extensions : Service for user and constrained delegation protocol. http://msdn.microsoft.com/en-us/library/cc246071.aspx.
11. Microsoft. Kerberos protocol registry entries and kdc configuration keys in windows server 2003. http://support.microsoft.com/kb/837361.
12. Microsoft. Ms-kile - initial population of the pac. http://msdn.microsoft.com/en-us/library/ee896835.aspx.
13. Microsoft. Ms-kile - supported encryption types bit flags. http://msdn.microsoft.com/en-us/library/ee808210.aspx.
14. Microsoft. Ms-nrpc - netlogon remote protocol. http://msdn.microsoft.com/en-us/library/cc237068.aspx.
15. Microsoft. Ms-pac - privilege attribute certificate data structure. http://msdn.microsoft.com/en-us/library/cc237948.aspx.
16. Microsoft. Network security : Restrict ntlm : Ntlm authentication in this domain. http://technet.microsoft.com/en-us/library/jj852241.aspx.
17. Microsoft. Planning for active directory forest recovery. http://technet.microsoft.com/fr-fr/library/planning-active-directory-forest-recovery.aspx.
18. Microsoft. Protected users security group. http://technet.microsoft.com/en-us/library/dn466518.aspx.
19. Microsoft. Public key cryptography for initial authentication (pkinit) in kerberos protocol. http://msdn.microsoft.com/en-us/library/cc238455.aspx.
20. Microsoft. Security account manager remote protocol (supplemental credentials structures). http://msdn.microsoft.com/en-us/library/cc245499.aspx.
21. Microsoft. Security_impersonation_level enumeration. http://msdn.microsoft.com/en-us/library/windows/desktop/aa379572.aspx.
22. Microsoft. Security_logon_session_data structure. http://msdn.microsoft.com/en-us/library/windows/desktop/aa380128.aspx.
23. Microsoft. User-account-control attribute. http://msdn.microsoft.com/en-us/library/ms680832.aspx.
24. Microsoft. Windows server 2012 : Planning for active directory forest recovery. http://www.microsoft.com/en-us/download/details.aspx?id=16506.

25. Microsoft. You experience a delay in the user-authentication process when you run a high-volume server program on a domain member in windows 2000 or windows server 2003. http://support.microsoft.com/kb/906736.

26. Jean-Yves Poublan. S4u2self et renforcement des services. http://blogs.msdn.com/b/jean-yves_poublan/archive/2007/04/08/s4u2self-et-renforcement-des-services.aspx.

27. Microsoft Open Specifications Support Team. Encryption type selection in kerberos exchanges. http://blogs.msdn.com/b/openspecification/archive/2010/11/17/encryption-type-selection-in-kerberos-exchanges.aspx.

28. Microsoft Open Specifications Support Team. Notes on kerberos kvno in windows rodc environment. http://blogs.msdn.com/b/openspecification/archive/2011/05/11/notes-on-kerberos-kvno-in-windows-rodc-environment.aspx.

29. Microsoft Open Specifications Support Team. To kvno or not to kvno, what is the version ! ? http://blogs.msdn.com/b/openspecification/archive/2009/11/13/to-kvno-or-not-to-kvno-what-is-the-version.aspx.

30. Microsoft Open Specifications Support Team. Understanding microsoft kerberos pac validation. http://blogs.msdn.com/b/openspecification/archive/2009/04/24/understanding-microsoft-kerberos-pac-validation.aspx.

31. Microsoft Open Specifications Support Team. Windows configurations for kerberos supported encryption type. http://blogs.msdn.com/b/openspecification/archive/2011/05/31/windows-configurations-for-kerberos-supported-encryption-type.aspx.