L'administration en silo

Aurélien Bordes
aurelien26@free.fr

Résumé. Cet article traite de la sécurité de l'authentification dans les environnements Active Directory. L'un des principaux problèmes survient lorsque les utilisateurs, et en particulier les administrateurs, s'authentifient : ils « mettent en danger » leur secret d'authentification. Afin d'éviter cette situation, il est nécessaire de définir des périmètres d'exposition de ces secrets et de s'assurer de l'isolation des périmètres.
Les bénéfices de l'authentification Kerberos seront donc étudiés ainsi que les nouveaux mécanismes de sécurité apparus dans les systèmes Windows ces dernières années : les revendications, le blindage Kerberos *(Kerberos Armoring)*, l'authentification composée *(Compound Authentication)*, les stratégies d'authentification et les silos d'authentification.
La compréhension et la mise en place de ces mécanismes permettront d'améliorer la sécurité des secrets d'authentification dans les environnements Active Directory.

1 Introduction

L'Active Directory n'a cessé, depuis son introduction en 1999, de gagner en importance au sein des systèmes d'information. De nos jours, il n'est pas rare de rencontrer des réseaux avec un ou plusieurs annuaires Active Directory auxquels sont rattachées la quasi-totalité des postes utilisateur, des serveurs et parfois des ressources industrielles. Conçus pour gérer des millions d'objets, certains annuaires Active Directory, par le jeu de rachat ou de fusion d'entreprises (et donc par la mise en place de relation d'approbation), peuvent gérer plusieurs centaines de milliers d'utilisateurs et d'ordinateurs.

De par ce rôle central, la sécurité de l'Active Directory est devenue un élément crucial. Signe des temps, les présentations sur cette thématique sont de plus en plus nombreuses. En effet, la prise de contrôle, par un attaquant, d'un annuaire Active Directory lui permet de disposer immédiatement d'une capacité d'administration de toutes les ressources du système d'information. Cela lui offre la possibilité d'accéder, en toute discrétion, à quasiment toutes les données métier et d'assurer aisément sa persistance.

Avec cette importance croissante, les pratiques d'administration ont dû évoluer. Initialement, et par facilité, tous les administrateurs étaient membres du groupe intégré « Administrateurs du domaine » même si cela n'était pas nécessaire. En effet, être membre de ce groupe donne tous les droits d'administration sur l'intégralité des systèmes et évite une fastidieuse délégation de droits. Mais la compromission par un attaquant d'un seul compte membre de ce groupe lui donne tous les droits. Il n'est pas rare d'avoir également des comptes de service membres du groupe « Administrateurs du domaine » et utilisés sur de nombreux serveurs voire sur les postes de travail.

Ainsi, des secrets d'authentification de niveau « Administrateurs de domaine » se voient disséminés un peu partout sur le système d'information. Ces secrets prennent la forme d'un mot de passe stocké (dans le Registre, dans un fichier, etc.), de secrets en mémoire (empreinte NTLM ou clés Kerberos) ou de traces réseau (élément de défi/réponse, bloc chiffré, etc.). Un attaquant a donc de multiples possibilités de récupérer un compte très privilégié sur le système d'information.

Face à ces problèmes, Microsoft propose plusieurs solutions techniques dont les dernières permettent de restreindre et de contrôler très finement les autorisations d'authentification des comptes de l'Active Directory. Cet article se propose d'étudier ces mécanismes de protection.

2 Niveau d'administration

Au sein d'un système d'information, les acteurs peuvent être répartis en trois catégories :

- **les ressources** qui hébergent les données métier (fichiers, messagerie, bases de données, etc.) ou les composants de gestion du système d'information (base de comptes utilisateur, contrôle d'accès, etc.). Par simplification, les ressources sont généralement représentées par les serveurs qui les hébergent[1] ;
- **les utilisateurs** qui accèdent à une partie des ressources. Les accès doivent être autorisés et peuvent être journalisés sans que l'utilisateur ne puisse agir sur le contrôle d'accès ou la journalisation, car, par définition, un utilisateur ne peut pas élever ses droits ou ses privilèges sur une ressource donnée ;

[1] Puisque qu'en général, la prise de contrôle d'un serveur (être administrateur de celui-ci) permet l'accès aux ressources hébergées sur celui-ci.

– **les administrateurs** : ce sont des utilisateurs qui, de par les droits ou privilèges qui leur sont conférés, disposent de pouvoirs particuliers qui permettent, en autre :
  • de gérer les autorisations d'accès aux ressources, ce qui leur permet implicitement d'accéder à plus de ressources que leur compte utilisateur (voire à toutes les ressources dans certains cas),
  • de pouvoir contourner certains mécanismes de sécurité (contrôle d'accès, journalisation, etc.),
  • de modifier la configuration des systèmes ou d'accéder à certaines informations sensibles (journal de sécurité, base de comptes, mémoire de processus, installation de *keylogger*, etc.).

Afin de segmenter les niveaux d'administration, il est nécessaire au préalable de segmenter les ressources. Le modèle proposé est le suivant :

– les ressources et les serveurs hébergeant les données métier (messagerie, fichiers, base de données, etc.) ou des serveurs d'infrastructure (sauvegarde, antivirus, WSUS, SCOM, etc.) sont positionnés à un niveau appelé **JAUNE**. Ce niveau comporte, en première analyse, tous les serveurs ;
– les ressources et les serveurs hébergeant des mécanismes d'administration auxquels toutes les ressources des autres niveaux sont adhérentes sont positionnés à un niveau appelé **ROUGE**. De par leur nature, la prise de contrôle d'une ressource de ce niveau permet de contrôler toutes les ressources, y compris celles des autres niveaux. Ce niveau comporte, en première analyse, les contrôleurs de domaine ;
– les postes de travail et le reste sont quant à eux positionnés dans un niveau appelé **VERT**. Il s'agit du niveau le plus exposé aux attaquants.

La segmentation des administrateurs vient respecter celle des niveaux des ressources, ce qui donne :

– niveau ROUGE : les administrateurs des contrôleurs de domaine ou les administrateurs ayant la possibilité de récupérer ou de changer le mot de passe (ou plus généralement un secret d'authentification) de l'ensemble des comptes (utilisateur ou machine) du domaine. Il inclut, en première analyse, tous les membres, directs ou par imbrication, du groupe du domaine « Administrateurs »[2]. Pour rappel, le groupe « Administrateurs du domaine » est membre de ce groupe ;

[2] Bien que ce groupe soit un groupe du domaine, l'appartenance à ce groupe ne confère des droits d'administration que sur les contrôleurs de domaine (au sens système d'exploitation).

- niveau JAUNE : les administrateurs des serveurs. Il s'agit des administrateurs ayant des capacités d'administration sur une partie ou la totalité des serveurs et des ressources associées. Par défaut, il contient tous les comptes (du domaine ou locaux) étant membres des groupes des administrateurs locaux des serveurs. Attention ici à ne pas confondre avec le groupe intégré du domaine « Opérateurs de serveur », qui est de niveau ROUGE (voir ci-dessous) ;
- niveau VERT : les administrateurs des postes de travail. Il contient tous les comptes (du domaine ou locaux) étant membres des groupes des administrateurs locaux des postes de travail. Attention ici à ne pas confondre avec les personnes d'un *Helpdesk* qui ne doivent pas avoir de droit particulier sur les postes de travail et qui agissent dans le contexte de sécurité des utilisateurs *via* un logiciel de prise de main à distance (par exemple avec MSRA). De par l'exposition des postes de travail, ce sont en général ces comptes d'administration qui sont compromis en premier lors d'une attaque.

Quant aux utilisateurs, ils n'ont, par définition, pas de segmentation. Un utilisateur peut utiliser n'importe quelle ressource et ne doit pas, par son utilisation, pouvoir en prendre le contrôle. Le meilleur exemple est celui des contrôleurs de domaine qui sont de niveau ROUGE mais dont les services (SMB, LDAP, RPC, etc.) doivent être utilisés et accessibles par tous les utilisateurs du domaine.

La répartition proposée est évidemment théorique en particulier la répartition entre le niveau JAUNE et VERT, forcément liée à la complexité du système d'information. Il est indispensable d'étudier le périmètre exact des ressources et de déterminer les acteurs[3] ayant des droits d'administration pour chaque niveau. Il faut notamment s'assurer que les ressources et les administrateurs d'un niveau donné ne peuvent être administrés, de manière effective, que par des acteurs d'un même niveau. En particulier, il ne faut pas se limiter aux membres des groupes. Toute la difficulté réside dans la capacité à déterminer exhaustivement tous les membres d'un niveau donné.

À titre d'exemple, concernant le niveau ROUGE, cela donne, en plus des membres des groupes « Administrateurs » et « Administrateurs du domaine » :

- les contrôleurs de domaine (qui hébergent la base des comptes). Par exemple, si un attaquant à la possibilité de modifier le mot de passe

[3] Le terme « acteur » fait référence ici à un *Principal* au sens Active Directory, c'est-à-dire un compte (utilisateur, de service ou de machine) pouvant s'authentifier et donc disposer d'un contexte de sécurité.

d'un compte du domaine associé à un contrôleur de domaine (parce qu'il contrôle un compte étant propriétaire de l'objet dans l'Active Directory), il est en mesure d'effectuer une réplication et de récupérer toutes les informations des comptes, y compris les « attributs secrets[4] » ;
- les postes d'administration sur lesquels les administrateurs ROUGE s'authentifient et réalisent leurs tâches ;
- les serveurs WSUS depuis lesquels les contrôleurs de domaine ou les postes d'administration ROUGE récupérèrent leurs mises à jour ;
- les serveurs de sauvegarde de la base de l'Active Directory ;
- les serveurs de gestion de parc (type SCOM) auxquels les contrôleurs de domaine sont rattachés ;
- les autorités de certifications approuvées pour l'authentification Windows au sein de la forêt[5] ;
- les groupes intégrés d'administration :
  - « Administrateurs de l'entreprise » qui possèdent des droits sur le domaine racine de la forêt,
  - « Administrateurs du schéma » qui peuvent notamment modifier les descripteurs de sécurité par défaut de tous les objets de l'annuaire y compris ceux sensibles (compte utilisateur, groupe, GPO, etc.),
  - « Opérateurs de serveur » qui peuvent gérer les services Windows, donc exécuter du code (y compris sur les contrôleurs de domaine),
  - « Opérateurs de sauvegarde » qui peuvent accéder, à distance, à n'importe quel fichier,
  - « Opérateurs de compte » qui peuvent gérer tous les comptes utilisateur, à l'exception de ceux protégés par l'`AdminSdHolder` ;
- tous les comptes d'administration qui gèrent des ressources dont la prise de contrôle permet la compromission de ressources du niveau ROUGE (SAN, Hyperviseurs, etc.).

3 Isolation des niveaux

Afin que la classification par niveau d'administration soit pertinente, il faut s'assurer de « l'imperméabilité » des niveaux entre eux. Ceci consiste à s'assurer de l'impossibilité, pour un acteur d'un niveau VERT ou JAUNE,

[4] Les attributs secrets sont ceux qui herbergent des données très sensibles comme les empreintes des mots de passe, les clés Kerberos et les historiques des secrets. Ils ne peuvent pas être récupérés par requête LDAP.

[5] Voir, à ce titre, les autorités de certification approuvées dans le conteneur `CN=NTAuthCertificates,CN=Public Key Services,CN=Services` de la partition de configuration de la forêt.

de prendre le contrôle d'une ressource ou d'un compte d'administration d'un niveau supérieur.

Pour gagner la prise de contrôle de comptes d'administration, un attaquant va mettre en œuvre plusieurs scénarios d'attaque. Une méthode généralement utilisée est de rechercher, puis de réutiliser, des secrets d'authentification *(Credentials)*. En effet, les administrateurs, de par leur activité, se connectent à de nombreux systèmes ce qui a pour effet de « disséminer » leur secret d'authentification un peu partout sur le système d'information. La mise en place des niveaux d'administration doit permettre de délimiter, pour chaque niveau, la frontière de diffusion des secrets d'authentification des administrateurs.

Évidemment, l'isolation du niveau ROUGE est la plus importante : même si ce niveau ne comporte pas de ressource hébergeant des données, sa prise de contrôle par un attaquant est désastreuse et permet à celui-ci d'accéder à quasiment toutes les ressources de tous les niveaux. De plus, le niveau ROUGE confère également des droits d'administration locaux aux systèmes Windows, ce qui augmente la furtivité et les moyens de persistance.

Toutefois, le niveau JAUNE ne doit surtout pas être oublié. Il contient en effet l'ensemble des données métier. Sa compromission permet de réaliser de l'espionnage industriel massif ainsi que des attaques destructrices importantes. Si la compromission du niveau ROUGE est souvent recherchée par les attaquants par facilité (car il procure un accès immédiat et privilégié à toutes les ressources, et ce, avec un haut niveau de furtivité), la compromission des données et des informations contenues au niveau JAUNE reste toujours l'objectif final.

La suite de l'article va s'intéresser aux risques de dissémination des secrets d'authentification et aux moyens de protection et d'isolation pouvant être mis en œuvre afin d'assurer l'imperméabilité entre les niveaux.

4 Sécurité de l'authentification, de NTLM à Kerberos

La dissémination des secrets d'authentification est liée au protocole utilisé lors des opérations d'authentification. Cette partie étudie les deux principaux protocoles supportés par Windows et les risques associés.

4.1 NTLM

L'authentification NTLM est le protocole d'authentification historique de Windows. Celui-ci repose sur un classique défi/réponse. Le synopsis

classique d'authentification d'un compte de domaine depuis un client vers un système membre est le suivant (voir figure 1) :

- l'utilisateur saisit son mot de passe sur son poste (①) ;
- celui-ci est converti en empreinte NTLM et est éventuellement conservé par LSASS pour les besoins du SSO de Windows (②) ;
- lorsque l'utilisateur souhaite s'authentifier auprès d'un service distant, NTLMSSP est mis en œuvre :
  - un défi est envoyé par le serveur (③),
  - à partir de l'empreinte NTLM, une réponse est calculée et envoyée au serveur (④) ;
- s'il s'agit d'un compte du domaine, le système membre ne peut pas valider directement la réponse car il ne dispose pas des empreintes NTLM des comptes du domaine. Le système membre valide donc le défi/réponse en les envoyant à un contrôleur de domaine *via* le protocole NETLOGON (⑤).

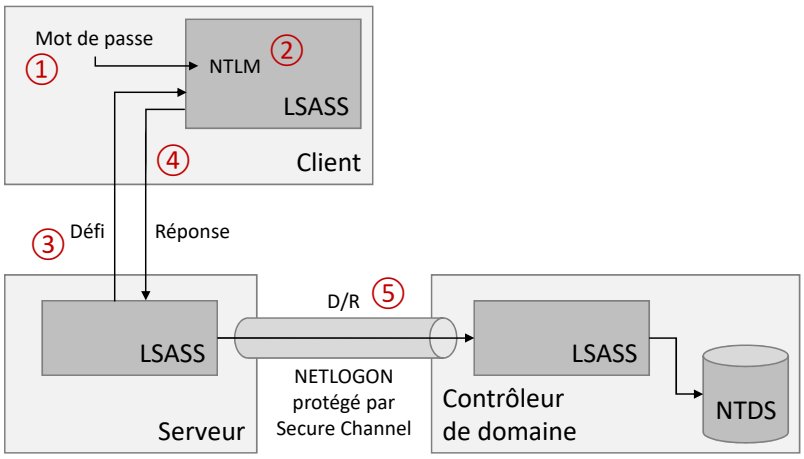

Fig. 1. Synopsis de l'authentification NTLM

Cette forme d'authentification souffre de plusieurs vulnérabilités dont la principale réside dans le fait que la réponse est directement calculée depuis le secret d'authentification de l'utilisateur (l'empreinte NTLM). Ainsi, si un attaquant arrive à récupérer un couple défi/réponse, il sera en mesure de mener une attaque en force brute sur le mot de passe. Ceci est d'autant plus aisé que NTLM met en œuvre des algorithmes performants pour les calculs cryptographiques (MD4 et HMAC-MD5) et n'utilise pas de graine. Le défi/réponse est donc à considérer comme faisant partie des secrets d'authentification de l'utilisateur.

L'authentification NTLM protège assez mal les secrets. Il est ainsi possible de les retrouver (voir figure 2)[6] :

- sur le poste du client, où le mot de passe est accessible lors de sa saisie et où l'empreinte NTLM[7] est mise en cache ;
- au niveau du réseau, où le défi/réponse circule en clair ;
- sur le serveur, où le défi/réponse peut-être intercepté. Ce point est très important car il signifie qu'avec NTLM, **quand un utilisateur s'authentifie auprès d'un système distant, il expose ses secrets d'authentification**.

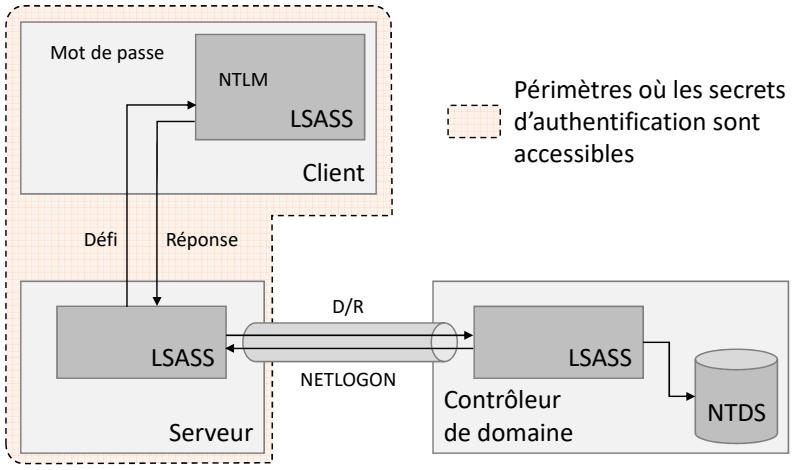

Fig. 2. Frontières de diffusion des secrets avec NTLM

En revanche, l'accès aux échanges réseau entre le serveur et les contrôleurs de domaine ne permettent pas d'accéder aux secrets d'authentification. En effet, NETLOGON est protégé par *Secure Channel* qui chiffre les informations avec les clés des comptes machine.

De plus, NTLM est vulnérable à d'autres formes d'attaque, en particulier le *NTLM Credentials Forwarding*. Cette attaque consiste à intercepter les défis/réponses d'une authentification pour les rejouer auprès d'un tiers :

- un attaquant souhaite s'authentifier sur un serveur ;
- celui-ci lui envoie un défi ;

[6] Évidemment, les contrôleurs de domaine doivent également être inclus dans le périmètre car ils gèrent la base des comptes. Mais ceux-ci étant de niveau ROUGE, ils sont exclus par simplification.

[7] Pour rappel, l'empreinte NTLM est suffisante pour pouvoir s'authentifier à distance avec la technique du *Pass-the-Hash*.

– ce défi du serveur est soumis à une victime qui envoie sa réponse ;
– la réponse de la victime est envoyée au serveur ce qui permet à l'attaquant de s'authentifier au nom de la victime.

En conclusion, avec NTLM, l'exposition des secrets d'authentification est maximale : ceux-ci se retrouvent sur le poste du client, sur le serveur et circulent de manière non protégée sur le réseau entre le client et serveur.

4.2 Kerberos

Note : la suite de l'article nécessite une bonne connaissance du protocole Kerberos. Le lecteur est invité à lire un précédent article sur le sujet [2] qui présente plus en détail ce protocole. De plus, afin d'alléger les actes du SSTIC, les analyses complètes des échanges ne sont pas reproduites dans cet article mais sont disponibles en ligne [1]. Elles permettent de comprendre plus aisément les différents échanges et structures de données évoqués par la suite.

Kerberos, apparu avec Windows 2000, vise à être le protocole nominal d'authentification dans les environnements Active Directory. Kerberos change radicalement la forme de l'authentification *via* la mise en œuvre de tickets et de clés de session. Le synopsis classique d'authentification d'un compte de domaine est le suivant (voir figure 3) :

– l'utilisateur saisit son mot de passe (①) ;
– celui-ci est converti en clé Kerberos K_C[8] de l'utilisateur et est éventuellement conservé par LSASS pour les besoins du SSO de Windows (②) ;
– un TGT est demandé au KDC par le client (③) *via* un échange `AS_REQ`/`AS_REP`. Dans la requête, le client doit généralement se « pré-authentifier » en incluant un bloc `PA-ENC-TS-ENC` chiffré par K_C et qui contient la date courante ;
– un ticket de service est demandé au KDC par le client (④) *via* un échange `TGS_REQ`/`TGS_REP`. Dans la requête `TGS_REQ`, le client doit s'authentifier en incluant un message `AP_REQ`. L'authentification a lieu avec le TGT de l'utilisateur et le bloc `Authenticator` doit être correctement chiffré avec $S_{C,K}$;
– le client s'authentifie auprès d'un service *via* un échange de message `AP_REQ`/`AP_REP` (⑤) et en présentant un ticket de service et en chiffrant correctement avec $S_{C,S}$ le bloc `Authenticator`.

[8] En réalité, ce n'est pas une seule clé qui est calculée, mais plusieurs types de clés permettant d'être en mesure d'utiliser différents algorithmes (MD5/DES, MD4/RC4, SHA-1/AES, etc.).

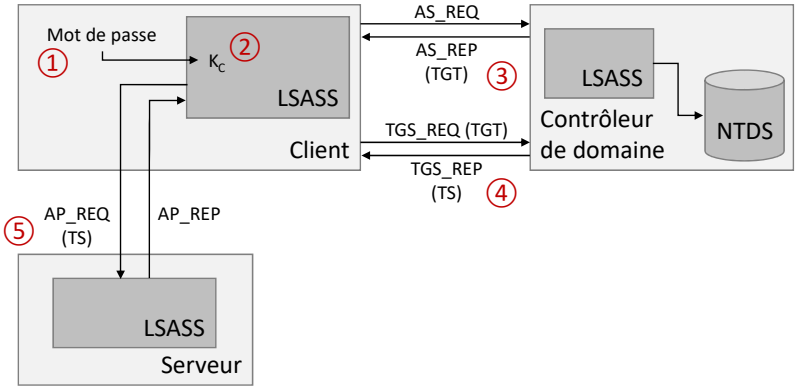

Fig. 3. Synopsis de l'authentification Kerberos

Une analyse de sécurité montre que les éléments à risque liés à une authentification d'un utilisateur sont :

- le poste du client, où le mot de passe et les clés Kerberos[9] sont accessibles ;
- le réseau entre le poste du client et le KDC, où transitent les messages `AS_REQ` et `AS_REP`. Ces deux messages sont sensibles car ils contiennent des éléments chiffrés directement par la clé Kerberos K_C de l'utilisateur (voir ci-dessous) ;
- le serveur, si une délégation d'authentification est mise en œuvre, qui permet :
  - de récupérer un TGT et la clé de session associée dans le cadre de la délégation complète,
  - de demander un ticket de service au nom de l'utilisateur dans le cadre de la délégation contrainte *(S4U2proxy)*. Cette situation peut être aggravée si la transition de protocole est autorisée : le service peut demander un ticket de service au nom de l'utilisateur sans que celui-ci ne se soit authentifié préalablement auprès du service *(S4U2self)*.

La clé K_C, qui peut être de différents types (DES-CBC-MD5, MD4 ou AES128/256-CTS-HMAC-SHA1-96), est calculée depuis le mot passe de l'utilisateur. Cette clé est utilisée en particulier pour chiffrer des blocs dans les messages `AS_REQ` ou `AS_REP` :

- dans `AP_REQ` : le bloc de pré-authentification `PA-ENC-TS-ENC` qui contient l'heure actuelle du client (`patimestamp`) ;

[9] Pour rappel, la clé Kerberos NTLM est suffisante pour pouvoir s'authentifier à distance avec la technique du *Pass-the-Key*.

– dans `AP_REP` : le bloc `EncKDCRepPart`, contenu dans la partie chiffrée de la réponse (`enc-part`), qui contient la clé de session $S_{C,K}$ partagée entre le KDC et le client.

Si ces blocs sont capturés, il est possible pour un attaquant de mener des attaques afin de retrouver le mot de passe de l'utilisateur. Cette situation est aggravée lorsque le type de clé est MD4 car, dans ce cas, les attaques en force brutes sont très performantes[10]. En revanche, les messages `TGS_REQ`/`TGS_REP` et `AP_REQ`/`AP_REP` ne contiennent que des blocs chiffrés par des clés de session générées aléatoirement. Leur capture ne met donc pas en danger les secrets d'authentification de l'utilisateur.

Kerberos renforce donc considérablement la protection des secrets d'authentification (voir figure 4). En particulier, si la délégation n'est pas autorisée, **un utilisateur n'expose pas ses secrets d'authentification quand il s'authentifie auprès d'un système distant**.

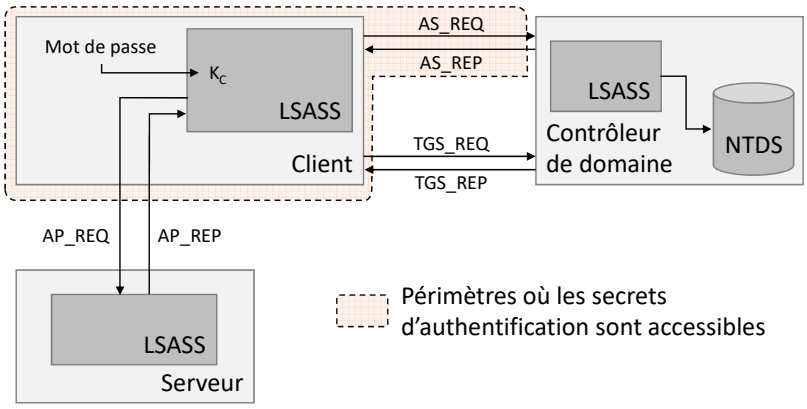

Fig. 4. Frontières de diffusion des secrets avec Kerberos

Kerberos change donc profondément la manière dont les secrets d'authentification doivent être protégés. Si la délégation n'est pas autorisée, un utilisateur peut se connecter à n'importe quel ordinateur sans risquer de compromettre ses secrets d'authentification. Avec Kerberos, la sécurité repose donc sur le système depuis lequel l'utilisateur s'authentifie.

5 Restrictions par les droits d'authentification

Afin d'empêcher des administrateurs de s'authentifier sur des ressources d'un niveau inférieur, il est possible d'utiliser les droits d'authentification

[10] Les clés MD4 sont, par construction, identiques à l'empreinte NTLM.

(Logon Rights) pour restreindre les autorisations d'authentification. Ce mécanisme à l'avantage d'être disponible sur toutes les versions de Windows. Pour rappel, les droits d'authentification sont au nombre de cinq, chacun séparé en un droit d'autorisation et un droit d'interdiction (voir tableau 1).

Droit d'autorisation **Droit d'interdiction**	**Description du droit d'authentification**
SeInteractiveLogonRight SeDenyInteractiveLogonRight	Log on locally
SeRemoteInteractiveLogonRight SeRemoteInteractiveLogonRight	Allows a user to log on to the computer using a Remote Desktop connection
SeNetworkLogonRight SeDenyNetworkLogonRight	Access this computer from network
SeServiceLogonRight SeDenyServiceLogonRight	Log on as a service
SeBatchLogonRight SeDenyBatchLogonRight	Log on as a batch job

Tableau 1. Droits d'authentification

Dans une configuration par défaut, les droits d'authentification autorisent tous les « Utilisateurs » locaux (S-1-5-32-545) à ouvrir une session locale (`SeInteractiveLogonRight`) et à s'authentifier par le réseau (`SeNetworkLogonRight`). Quant à l'accès par le « Bureau à distance » (`SeRemoteInteractiveLogonRight`), il n'est autorisé qu'aux membres des groupes « Administrateurs » (S-1-5-32-544) et « Utilisateurs du bureau à distance » (S-1-5-32-555).

Les droits d'interdiction étant prioritaires sur les droits d'autorisation, il suffit, pour interdire toute forme d'authentification d'un utilisateur ou d'un groupe d'utilisateurs, de rajouter son identifiant de sécurité dans les listes des cinq droits d'interdiction.

Pour faciliter le déploiement, les interdictions d'authentification peuvent être déployées par GPO afin d'interdire un groupe d'utilisateurs sur une partie des ordinateurs du domaine. Par exemple, il est souhaitable d'interdire l'authentification des membres du groupe « Administrateurs du domaine » sur tous les postes de travail (hors postes d'administration).

Malheureusement, la restriction d'authentification au moyen des droits d'authentification se révèle vite compliquée à configurer. En effet, les droits définis par dans des GPO ne se surchargent pas, rendant impossible la mise en œuvre d'une configuration fine. Ainsi, une interdiction configurée à une unité d'organisation (OU) donnée peut se voir annuler par une GPO

appliquée à une OU fille. Pour éviter cette situation, il faut que l'OU fille reprenne toutes les configurations des OU parentes et les fusionne avec ses propres paramètres.

Avec l'authentification NTLM, les restrictions d'authentification vues ci-dessus perdent de leur efficacité. En effet, les restrictions sont appliquées après validation de l'authentification. Si un attaquant contrôle une machine sur laquelle un administrateur s'authentifie (par inadvertance ou parce qu'il y est forcé), l'attaquant peut récupérer des secrets d'authentification avant que l'interdiction soit notifiée à l'utilisateur : un mot de passe pour une authentification interactive ou par le bureau à distance, un défi/réponse pour une authentification réseau.

6 Groupe *Protected users*

Le groupe « *Protected users*[11] », apparu avec Windows Server 2012 R2, est un dispositif primordial pour la protection des secrets d'authentification. L'appartenance à ce groupe implique que ses membres sont soumis à plusieurs protections dont les plus importantes sont :

- seule l'authentification par Kerberos est autorisée ;
- les clés K_C des utilisateurs ne sont pas mises en cache (seulement un TGT d'une durée de validité de 4 heures) ;
- seuls les algorithmes de chiffrement basés sur AES sont autorisés. Pour rappel, les clés AES sont calculées via une dérivation de clés basée sur PBKDF2[12] qui rend les attaques en force brute bien plus coûteuses ;
- toute forme de délégation d'authentification est interdite.

Comme vu dans la section 4.2, l'utilisation exclusive de Kerberos améliore grandement la protection des secrets d'authentification, les membres de ce groupe n'exposant pas leur secret d'authentification lors de l'authentification auprès d'un service. Ceci est d'autant plus vrai que la délégation d'authentification est interdite. De plus, l'utilisation de clés de type AES et la désactivation des clés de type RC4 renforcent la sécurité des messages `AS_REQ` et `AS_REP`, sans toutefois assurer totalement leur sécurité.

Cependant, peu d'applications supportent exclusivement l'authentification par Kerberos, ce qui limite la mise en œuvre du groupe *Protected users*. C'est heureusement le cas pour tous les outils d'administration de l'Active Directory. Il faut donc ajouter à ce groupe tous les administrateurs de niveau ROUGE (par exemple en ajoutant tous les groupes

[11] Ce groupe se caractérise par le RID `DOMAIN_GROUP_RID_PROTECTED_USERS`.
[12] Avec 4096 tours par défaut, cette valeur étant paramétrable.

« Administrateurs » ou « Opérateurs »). Ceci est un prérequis pour les protections détaillées dans les parties suivantes.

Note : pour être pleinement efficace, ce groupe nécessite, côté client, des systèmes Windows 8.1/Server 2012 R2 et, côté KDC, des versions Windows Server 2012 R2. Cependant, les fonctionnalités côté client ont été rétro-portés par le correctif KB2871997. La création du groupe dans l'Active Directory nécessite une extension de schéma vers Windows Server 2012 R2, qui peut être réalisé quelque soit le niveau fonctionnel et la version de Windows des contrôleurs de domaine. Ceci permet déjà de bénéficier des protections sur le poste de travail disposant d'une version récente de Windows ou du correctif le cas échéant.

7 Les revendications

Les revendications *(Claims)* viennent étendre le modèle d'autorisation historique de Windows basé sur les SID (identifiants de sécurité). Apparues avec AuthZ et le modèle CBAC *(Claims-Based Access Control)*, les revendications ont été étendues à l'Active Directory avec Windows Server 2012.

Les revendications sont des attributs de sécurité de différents types (INT64, UINT64, chaîne de caractères, booléen) [5] qui peuvent prendre part au contrôle d'accès de Windows. Les jetons de sécurité *(Tokens)*, les descripteurs de sécurité et la PAC[13] *(Privilege Attribute Certificate)* sont alors étendus pour prendre en compte les revendications.

Il existe trois types de revendications : utilisateur, ressource et périphérique *(Device)* [14].

7.1 Revendications utilisateur

Les « **revendications utilisateur** » sont des revendications affectées à des *Principals* lorsqu'ils s'authentifient. Ces revendications peuvent être générées à partir de quatre types de source [8] :

[13] Pour rappel, une PAC est une structure de données qui sert à transporter les données d'authentification d'un *Principal* : SID du domaine, RID de l'utilisateur, RID de ses groupes d'appartenance, *ExtraSID*, etc. Cette structure, initialement conçue et utilisée par NETLOGON est désormais également utilisée dans les tickets Kerberos.

[14] Il existe un 4e type appelé « *Local Claims* ». Ces attributs sont utilisés en interne de Windows en particulier pour AppLocker (`APPID://`), l'identification unique de processus (`TSA://ProcUnique`) ou les services utilisateur (`WIN://SCMUserService`). Ils sont stockés dans le champ `pSecurityAttributes` des jetons de sécurité *(Tokens)* et affichés, par Process Hacker, dans l'onglet « *Attributes* » de la page des propriétés avancées d'un *Token*.

- `AD` : la revendication est générée depuis un attribut du compte dans l'Active Directory. L'attribut est référencé par son nom ;
- `Certificate` : la revendication est générée depuis un attribut d'un certificat X.509. L'attribut est référencé par son OID ;
- `TransformPolicy` : la revendication est générée depuis une *Claims Transform Rules* (CTR). Ceci est utilisé pour transformer les revendications issues d'une autre forêt ;
- `Constructed` : la revendication est générée automatiquement. La seule revendication de ce type actuellement définie est `AuthenticationSilo` (voir section 9.3) et il n'est pas possible d'en définir d'autres.

Une source de revendication utilisateur doit être définie par un objet de type `msDS-ClaimType` positionné dans le conteneur `CN=Claims Configuration, CN=Services` de la partition de configuration de la forêt. Les propriétés suivantes de cet objet caractérisent la revendication :

- l'origine de l'attribut (`msDS-ClaimSourceType`) : `AD`, `Certificate`, `TransformPolicy` ou `Constructed` ;
- le type de données de la revendication (`msDS-ClaimValueType`) (voir [5] pour les différences types possibles) ;
- si le type de l'origine est `AD`, l'attribut `msDS-ClaimAttributeSource` référence le nom de l'attribut Active Directory devant être converti en revendication ;
- la ou les classes d'objets pour lesquelles les revendications doivent être appliquées (`msDS-ClaimTypeAppliesToClass`). Pour l'instant, les revendications peuvent être associées aux objets de classe utilisateur (`Users`), ordinateur (`Computer`) et compte de service (`ms-DS-Managed-Service-Account` ou `ms-DS-Group-Managed-Service-Account`).

L'exemple suivant définit une revendication utilisateur, « description », générée depuis l'attribut de l'Active Directory du même nom :

```
Dn: CN=ad://ext/description:88d46537a8db15a9, ↩
    CN=Claim Types,CN=Claims Configuration,CN=Services,<configuration>
objectClass (3): top; msDS-ClaimTypePropertyBase; msDS-ClaimType;
name: ad://ext/description:88d46537a8db15a9;
displayName: description;
msDS-ClaimAttributeSource: CN=Description,CN=Schema,CN=Configuration,DC=<forest>;
msDS-ClaimIsSingleValued: FALSE;
msDS-ClaimIsValueSpaceRestricted: FALSE;
msDS-ClaimSourceType: AD;
msDS-ClaimTypeAppliesToClass (2):
  CN=User,CN=Schema,CN=Configuration,DC=test,DC=local;
  CN=Computer,CN=Schema,CN=Configuration,DC=test,DC=local;
msDS-ClaimValueType: 3;   // CLAIM_SECURITY_ATTRIBUTE_TYPE_STRING
```

Lorsqu'un membre de l'AD, dont le type est inclus dans la liste `msDS-ClaimTypeAppliesToClass`, s'authentifie, ses revendications utilisateur sont générées. Avec Kerberos, celles-ci sont transmises au client *via* la PAC contenue dans le champ `authorization-data` d'un ticket. Dans une PAC, les revendications utilisateur sont transportées par un bloc de type `PAC_CLIENT_CLAIMS_INFO`. L'exemple suivant présente une revendication utilisateur appelée « `description` », issue de l'AD (`ad://`) et valant « Administration » :

```
[PAC_INFO_BUFFER]
  [PAC_CLIENT_CLAIMS_INFO] Version: 1, Count: 1
    [0] Name: ad://ext/description:88d46537a8db15a9, Type: 3, Count: 1
      [0] Administration
```

Lorsqu'un système Windows reçoit un ticket avec une PAC contenant un bloc `PAC_CLIENT_CLAIMS_INFO`, les revendications contenues dans ce bloc sont mises dans la partie `UserClaims` du *Token* généré (`_TOKEN.pClaimAttributes.pUserSecurityAttributes`[15]). Si les revendications sont correctement incluses dans le *Token*, le SID `CLAIMS_VALID`[16] est ajouté dans la liste des SID de l'utilisateur (`_TOKEN.UserAndGroups`).

Lorsqu'il s'agit d'un *Token* d'un utilisateur, il est possible de voir les revendications utilisateur au moyen de la commande `whoami.exe`. Process Hacker permet également d'afficher les revendications utilisateur dans l'onglet « *Claims* » de la page des propriétés avancées d'un *Token*, sous la dénomination « *User claims* » (voir figure 9 page 37).

```
C:\>whoami /claims

Claim Name    Claim ID                                Flags Type   Values
============= ======================================= ===== ====== ===========
"description" ad://ext/description:88d46537a8db15a9       String "Administration"
```

Note : le terme « utilisateur » peut être trompeur. Lorsqu'un compte de machine ou compte de service s'authentifie, il peut également lui être affecté des revendications utilisateur. Le terme « utilisateur » fait référence ici à l'entité *(Principal)* réalisant l'authentification.

7.2 Revendications ressource

Les « **revendications ressource** » permettent d'associer des revendications à des « objets sécurisables » (fichier, objet AD, imprimante, etc.),

[15] Structure de type `AUTHZBASEP_SECURITY_ATTRIBUTES_INFORMATION`.
[16] S-1-5-21-0-0-0-497

c'est-à-dire pouvant disposer d'un descripteur de sécurité. Les revendications ressource sont stockées directement dans le descripteur de sécurité de l'objet et celui-ci peut définir plusieurs revendications, chacune prenant la forme d'une ACE[17] de type `SYSTEM_RESOURCE_ATTRIBUTE_ACE_TYPE` située dans la SACL du descripteur de sécurité de l'objet.

La structure de telles ACE est `SYSTEM_RESOURCE_ATTRIBUTE_ACE` [11]. Il s'agit simplement d'une ACE contenant une structure `CLAIM_SECURITY_ATTRIBUTE_RELATIVE_V1` [9] qui permet de caractériser une revendication. La fonction `AddResourceAttributeAce` [15] de l'API Windows permet de manipuler ce type d'ACE.

Par exemple, le descripteur de sécurité ci-dessous, défini une revendication ressource (`RA` pour `SDDL_RESOURCE_ATTRIBUTE`) appelée « `Confidentiality_MS` », de type entier signé (`TI` pour `SDDL_INT`) et valant `3000` (niveau *Medium*)[18]. L'éditeur graphique permet également d'afficher les revendications sur une ressource (voir figure 5 avec la revendication « *Confidentiality* »).

```
S:AI(RA;;;;;WD;("Confidentiality_MS",TI,0x20020,3000))
```

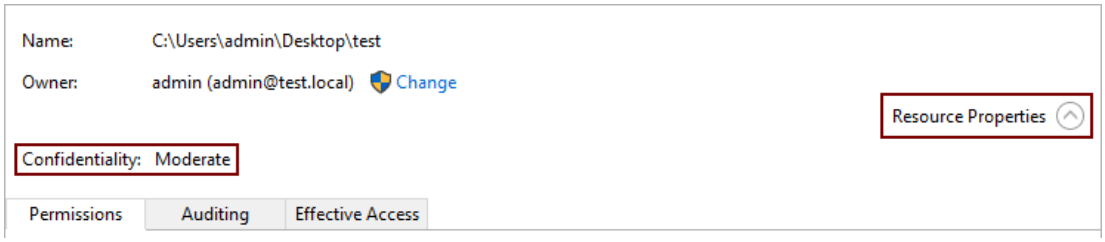

Fig. 5. Affichage des revendications ressource avec l'éditeur graphique

Dans la pratique, les revendications ressource sont rarement appliquées manuellement. Sous Windows Server, le service *File Server Resource Manager* permet, *via* le *Classification Management*, de définir des revendications et des règles de classification[19] pour les fichiers. Le service se charge alors d'appliquer périodiquement les règles et de mettre à jour les revendications ressource des fichiers.

[17] Une ACE *(Access Control Entry)* est une entrée dans une liste d'accès (DACL ou SACL).

[18] Ce type d'ACE doit toujours être appliqué à l'entité *Everyone* (`WD`).

[19] La définition des règles peut-être très avancée et reposer, par exemple, sur le nom du fichier, son contenu (analysé par des expressions rationnelles) ou des scripts PowerShell.

7.3 Revendications périphérique

Les « **revendications périphérique** » sont abordées dans la section 9.4 qui traite de l'« authentification composée ».

8 Conditional ACEs

L'extension du modèle de sécurité permet également de définir des droits d'accès basés sur les revendications. Ces nouveaux droits sont définis au moyen d'ACE appelées « *Conditional ACEs* » [20]. Ci-dessous un exemple de SDDL[21] d'un descripteur avec une ACE de ce type :

```
"D:PAI(XA;OICI;0x1301bf;;;WD;(@USER.ad://ext/description:88d46537a8db15a9 ↩
   Any_of "Administration", "Gestion"))"
```

Cet exemple de SDDL définit une ACE de type `SDDL_CALLBACK_ACCESS_ALLOWED` (`XA`) autorisant tout le monde (`WD`) en accès en modification (`0x1301bf`) à condition que l'entité qui demande l'accès possède une revendication utilisateur émise par l'AD (`@USER.ad`) appelée « description » et devant valoir « Administration » ou (`Any_of`) « Gestion ». La grammaire de ce type d'ACE est donnée en [10] et [16]. Celle-ci est plutôt ardue mais permet des opérations complexes mettant en œuvre les revendications.

9 Évolution de Kerberos

Avec Windows Server 2012, Microsoft a apporté à Kerberos plusieurs fonctionnalités importantes : le blindage Kerberos *(Kerberos Armoring)*, l'authentification composée *(Compound Authentication)*, les stratégies d'authentification et les silos d'authentification. Ces fonctionnalités, détaillées ci-dessous, visent essentiellement à améliorer la protection des secrets d'authentification afin d'en limiter la dissémination. Elles sont donc particulièrement utiles pour assurer l'étanchéité des frontières des niveaux d'administration abordées dans la section 3, en particulier celle du niveau ROUGE.

[20] Ces ACE sont un sous-type des ACE de type « *Callback ACE* ».

[21] Le SDDL *(Security Descriptor Definition Language)* est un langage permettant de représenter textuellement un descripteur de sécurité.

9.1 Blindage Kerberos (FAST)

Présentation Le blindage Kerberos est l'implémentation de la technologie FAST *(Flexible Authentication via Secure Tunneling)* [3]. FAST permet, *via* la définition de nouveaux types de blocs de pré-authentification et de méthode de calcul de clés, de renforcer la sécurité des messages `AS_REQ`/`AS_REP` et `TGS_REQ`/`TGS_REP`. En particulier, FAST permet de protéger tous les blocs qui dépendent directement du mot de passe de l'utilisateur et qui sont vulnérables à des attaques sur celui-ci (voir section 4.2).

Un autre intérêt du blindage Kerberos est de permettre au KDC, lorsqu'un compte utilisateur ou de service s'authentifie, de disposer du TGT de la machine depuis laquelle le compte réalise son authentification. Dans ce TGT se trouvent les données d'autorisation du « périphérique » *(Device)* du compte. Ceci est utilisé pour deux fonctionnalités :

- les **stratégies d'authentification** lors de la demande de TGT (requêtes `AS_REQ`) pour un compte utilisateur ou de service (voir section 9.2) ;
- l'**authentification composée** lors de la demande de ticket de service (requêtes `TGS_REQ`) pour un compte utilisateur ou de service (voir section 9.4) ;

Les comptes de machines ne peuvent pas bénéficier de ces deux mécanismes.

Gestion des clés Pour protéger les messages Kerberos, FAST définit une « clé de blindage » *(Armor key)* qui sert de clé de chiffrement ou d'intégrité pour plusieurs blocs de données. Le calcul de cette clé dépend du type de blindage mis en œuvre. Afin de permettre la dérivation de clés, FAST introduit une fonction appelée `KRB-FX-CF2` permettant de calculer une clé à partir de la combinaison de deux autres clés. Le prototype de cette fonction est le suivant :

```
protocol key = KRB-FX-CF2(protocol key, protocol key,
                          octet string, octet string)
```

Les algorithmes utilisés par `KRB-FX-CF2` dépendent des types de clés devant être combinées. Quant aux chaînes `octet string`, elles sont définies pour chaque opération de combinaison.

Note : dans la suite de l'article, le symbole $\oplus$ sera utilisé pour référencer la fonction `KRB-FX-CF2` de composition de clés. Évidemment, cette fonction est bien plus complexe qu'un simple XOR.

Note : avec Kerberos, la notation C (K_C, $S_{C,K}$) désigne le « Client », c'est-à-dire le compte réalisant l'authentification Kerberos. Il peut donc s'agit d'un compte utilisateur ou d'un compte machine. Dans la suite, afin de pouvoir dissocier les deux, la notation C représentera le compte utilisateur et C' le compte de sa machine.

Modification des messages AS_REQ Dans les échanges AS protégés par FAST, le blindage est obligatoire. Ainsi, le bloc de pré-authentification PA-ENC-TIMESTAMP doit être remplacé par un bloc PA-FX-FAST. Celui-ci contient un bloc PA-FX-FAST-REQUEST contenant une structure de type KrbFastArmoredReq composée de 3 champs :

- armor qui indique, par sa présence, que le blindage est mis en place pour le message *(Explicit armor)*. Pour les messages AS_REQ où le blindage est obligatoire, ce champ est toujours présent et il indique le type de blindage mis en œuvre. Pour l'instant, un seul type de blindage (FX_FAST_ARMOR_AP_REQUEST) est défini par la RFC ;
- req-checksum qui contient un *checksum* permettant de vérifier l'intégrité d'un élément de la requête. Pour un message AS_REQ, la vérification concerne le bloc KDC-REQ-BODY du champ req-body. Pour rappel, sans FAST, ce champ est en clair et sans protection d'intégrité. L'*Armor key* est utilisée comme clé d'authentification pour le calcul du *checksum* ;
- enc-fast-req qui contient une structure KrbFastReq chiffrée par l'*Armor key* et qui contient deux champs importants :
  - padata : qui contient des blocs de pré-authentification. Parmi ces blocs, il doit être présent, pour les échanges AS, un bloc d'authentification de type PA-ENCRYPTED-CHALLENGE. Celui-ci contient une structure PA-ENC-TS-ENC chiffrée par une clé appelée *Challenge key*. Cette structure, qui contient l'heure du client, joue un rôle équivalent au bloc de pré-authentification Kerberos lorsque FAST n'est pas utilisé,
  - req-body qui reprend les données du champ du même nom contenu dans le message AS_REQ. Cependant, le KDC doit préférer ce champ, car son chiffrement par l'*Armor key* lui assure une protection en confidentialité et intégrité, contrairement à celui du message AS_REQ.

Le blindage de type FX_FAST_ARMOR_AP_REQUEST *(Ticket-Based Armor)* repose sur une authentification Kerberos effectuée avec un message AP_REQ. Le ticket utilisé pour cette authentification est appelé l'*Armor*

ticket. De préférence[22], celui-ci est le TGT de la machine depuis laquelle l'utilisateur s'authentifie. Ceci nécessite donc que la machine dispose d'un compte dans l'Active Directory (ce qui est toujours le cas) et que la machine se soit authentifiée en Kerberos préalablement à l'authentification de l'utilisateur (ce qui est généralement le cas[23]).

Note : dans un message `AP_REQ`, il est possible d'échanger une nouvelle clé *via* le champ `subkey` du bloc de l'`authenticator`. Si ce champ est optionnel, il doit être présent dans un message `AP_REQ` utilisé pour un blindage.

L'*Armor key* est calculée avec la combinaison des clés suivantes[24] (voir figure 6) :

- `ticket_session_key` : clé de session $S_{C',K}$ contenue dans le TGT de la machine
- `subkey` : sous-clé présente dans l'`authenticator`

```
Armor key = KRB-FX-CF2(subkey, ticket_session_key, "subkeyarmor", "ticketarmor")
```

Quant à la *Challenge key*, elle est calculée avec la combinaison des clés suivantes :

- `armor_key` : *Armor key* calculée suivant le type de blindage retenu
- `long_term_key` : clé K_C de l'utilisateur

```
Challenge key = KRB-FX-CF2(armor_key, long_term_key,
                           "clientchallengearmor", "challengelongterm")
```

Lorsque le KDC reçoit un message `AS_REQ`, la présence du bloc `PA-FX-FAST` lui indique que la demande est blindée. Le KDC disposant des clés Kerberos de l'utilisateur et de la machine, il peut procéder au calcul des clés intermédiaires puis au déchiffrement et à la vérification de la requête. Les opérations effectuées sont les suivantes :

- détermination du blindage utilisé ;

[22] La RFC propose l'utilisation d'autres TGT (TGT anonyme avec PKINIT, avec ou sans authentification du KDC), mais les systèmes Windows n'utilisent que le TGT de la machine.

[23] La machine peut ne pas obtenir un TGT avant l'utilisateur si aucun KDC n'est joignable à son démarrage.

[24] La combinaison de clés permet de s'assurer que chacune des deux parties contribue au calcul de l'*Armor key* : le KDC pour la clé contenue dans le TGT et le client pour la clé contenue dans l'`authenticator`

– récupération et déchiffrement de l'*Armor ticket* (le TGT de la machine) ce qui permet de récupérer la clé de session associée ($\texttt{S}_{\texttt{C'},\texttt{K}}$) ;
– déchiffrement de l'`Authentificator` et récupération de la `subkey` ;
– calcul de l'*Armor key* ($\texttt{S}_{\texttt{C'},\texttt{K}} \oplus$ `subkey`) ;
– calcul de la *Challenge key* (`Armor key` $\oplus$ $\texttt{K}_\texttt{C}$) ;
– déchiffrement de la structure KrbFastReq avec l'*Armor key* ;
– vérification du bloc `PA-ENC-TS-ENC` avec la *Challenge key*.

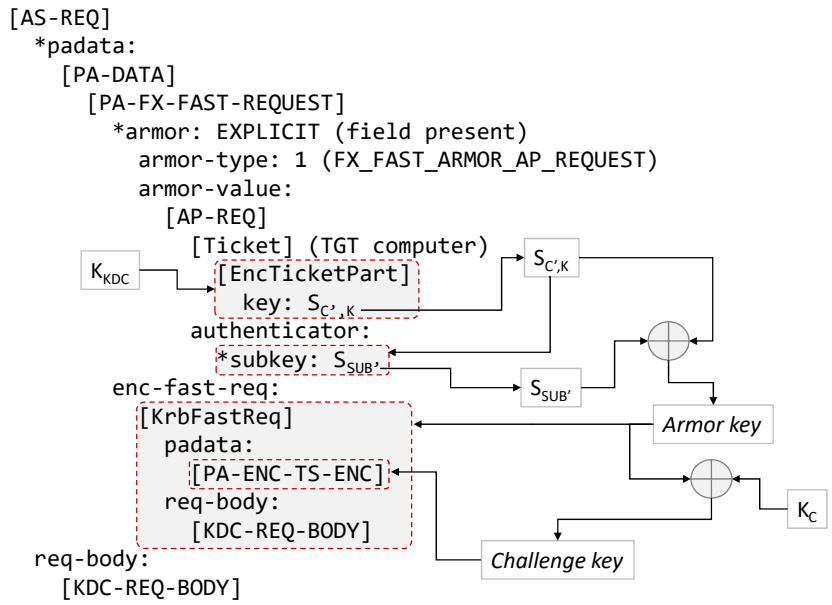

Fig. 6. Requête `AS_REQ` protégée par FAST

Modification des messages `AS_REP` Si la requête a été blindée, le KDC doit également blinder sa réponse. Ceci se traduit par la présence d'un bloc de type `PA-FX-FAST-REPLY` dans le champ `padata` du message `AS_REP`. Ce bloc doit être chiffré par l'*Armor key*[25] et contenir une structure de type `KrbFastResponse`. Comme pour l'authentification du client avec les messages `AS_REQ`, le KDC doit s'authentifier en prouvant qu'il peut calculer la *Challenge key* et chiffrer correctement un bloc `PA-ENCRYPTED-CHALLENGE`.

La *Challenge key* est calculée avec la combinaison des clés suivantes :

– `armor_key` : *Armor key* calculée suivant le type de blindage retenu
– `long_term_key` : clé $\texttt{K}_\texttt{C}$ de l'utilisateur

[25] L'*Armor key* utilisée par le KDC pour traiter la requête est reprise pour générer la réponse au client. Celui-ci procède de la même façon.

```
Challenge key = KRB-FX-CF2(armor_key, long_term_key,
                           "kdcchallengearmor", "challengelongterm")
```

L'authentification du KDC permet d'éviter certaines attaques qui consistent à forger des messages `AS_REP` uniquement à partir de la connaissance de K_C (de l'utilisateur). Si le KDC arrive à chiffrer correctement le bloc `PA-ENCRYPTED-CHALLENGE`, cela prouve qu'il connaît également la clé $K_{C'}$ du compte de la machine.

`KrbFastResponse` peut contenir une « clé de renforcement » *(Strengthen key)* dans le champ `strengthen-key`. Comme le nom le suggère, cette clé est générée aléatoirement par le KDC et permet de « renforcer » la clé utilisée pour chiffrer la partie chiffrée de la réponse `AS_REP` (champ `enc-part`). Sans FAST, c'est la clé K_C de l'utilisateur qui est utilisée, ce qui expose les secrets d'authentification de l'utilisateur en cas de capture réseau des messages `AS_REP` (voir section 4.2). La mise en œuvre[26] de la clé de renforcement permet de protéger les blocs qui sont chiffrés directement avec K_C et ainsi d'éviter les attaques sur le mot de passe de l'utilisateur. Le calcul de la clé de renforcement est effectué avec les clés suivantes :

- `strengthen-key` : clé spécifiée dans le champ *strengthen-key* de la structure `KrbFastResponse`
- `padata-reply-key` : clé K_C (pour les messages `AS_REP`)

```
reply-key = KRB-FX-CF2(strengthen-key, padata-reply-key,
                       "strengthenkey", "replykey")
```

Lorsque le client reçoit un message `AS_REP`, la présence du bloc `PA-FX-FAST` lui indique que la réponse est blindée. Le client procède alors au calcul des clés intermédiaires et au déchiffrement et à la vérification de la réponse en procédant aux opérations suivantes :

- détermination du blindage utilisé ;
- déchiffrement du bloc `KrbFastResponse` avec l'`Armor key` (la même que celle utilisée pour la requête) ;
- calcul de la *Challenge key* (`Armor key` $\oplus$ K_C) ;
- vérification du bloc `PA-ENC-TS-ENC` avec la *Challenge key* ;

[26] La RFC stipule que l'utilisation de la clé de renforcement est optionnelle pour les réponses AS mais obligatoire pour les réponses TGS. Les systèmes Windows utilisent systématiquement une clé de renforcement pour les réponses `AS_REP`, sauf dans le cas où PKINIT est mis en œuvre : dans ce cas, ce n'est K_C qui authentifie l'utilisateur, mais sa clé privée.

- récupération de la `strengthen-key` dans le bloc `KrbFastResponse` ;
- calcul de la `reply-key` renforcée (*strengthen-key* $\oplus$ $\mathtt{K_C}$) ;
- déchiffrement du bloc `EncKDCRepPart` avec la `reply-key` renforcée ;
- récupération de la clé de session associée au TGT ($\mathtt{S_{C,K}}$).

Apports de FAST pour les échanges AS Tout d'abord, il est important de remarquer que les échanges Kerberos `AS_REQ` et `AS_REP` des comptes de machine ne sont pas vulnérables aux attaques sur le mot de passe. En effet, ceux-ci sont générés aléatoirement avec une taille de 256 bits ce qui rend toute attaque sur le mot de passe théoriquement impossible. Le blindage Kerberos repose sur ce postulat.

Ensuite, concernant l'authentification des utilisateurs, FAST permet de protéger tous les blocs de données chiffrés avec $\mathtt{K_C}$. Les attaques vues dans la section 4.2 n'ont alors plus lieu d'être. Ainsi, avec FAST, lorsqu'un utilisateur s'authentifie, **le périmètre d'exposition de ses secrets est restreint au système depuis lequel l'authentification a lieu**.

Modification des messages `TGS_REQ` Le principe de mise en œuvre de FAST dans les échanges `TGS` est similaire à celui pour les échanges `AS`. Il existe cependant une variante importante : si le blindage est obligatoire pour les échanges AS, il est en revanche facultatif pour les échanges TGS. On parle alors de blindage implicite (« *Implicit armor* ») ou de blindage explicite (« *Explicit armor* »).

Dans tous les cas, lorsque FAST protège l'échange, un bloc `PA-FX-FAST` est présent dans les données de pré-authentification. Comme pour une requête `AS_REQ`, celui-ci contient un bloc `PA-FX-FAST-REQUEST` contenant une structure de type `KrbFastArmoredReq`.

La différence vient du champ `armor` de cette structure :

- si le champ `armor` n'est pas présent, le blindage est implicite ;
- si le champ `armor` est présent, un blindage explicite est mis en œuvre.

Avec un blindage implicite d'une requête `TGS_REQ`, seul le TGT de l'utilisateur est envoyé par le client. Il s'agit de celui présenté dans le message `AP_REQ` permettant à l'utilisateur de s'authentifier au service de délivrance des tickets. Le calcul de l'*Armor key* est alors effectué avec les clés suivantes (voir figure 7) :

- `ticket_session_key` : clé de session $\mathtt{S_{C,K}}$ contenue dans le TGT de l'utilisateur
- `subkey` : sous-clé présente dans l'`authenticator`

```
Armor key = KRB-FX-CF2(subkey, ticket_session_key, "subkeyarmor", "ticketarmor")
```

En revanche, avec un blindage explicite, le champ `armor` est présent. Celui-ci indique le blindage mis en œuvre : comme pour les requêtes `AS_REQ`, seul le type de blindage `FX_FAST_ARMOR_AP_REQUEST` est défini. Celui-ci repose sur un message `AP_REQ` avec comme *Armor ticket* le TGT de la machine. Le calcul de l'*Armor key* est, dans ce cas, opéré en deux étapes (voir figure 8). La 1re étape combine les clés de la requête `AS_REQ` de la machine (celle du blindage) :

- `ticket_session_key` : clé de session $S_{C',K}$ contenue dans le TGT de la machine
- `subkey` : sous-clé présente dans l'`authenticator` présenté avec le TGT de la machine

```
temp_key = KRB-FX-CF2(subkey, ticket_session_key, "subkeyarmor", "ticketarmor")
```

La 2e étape combine la sous-clé présente dans l'`authenticator` présenté avec le TGT de l'utilisateur avec la clé calculée ci-dessus avec :

```
Armor key = KRB-FX-CF2(subkey, temp_key, "subkeyarmor", "tgsarmor")
```

L'*Armor key* ainsi calculée, en implicite ou en explicite, sert pour la génération des éléments du bloc `KrbFastArmoredReq` :

- `req-checksum` qui contient un *checksum* où l'*Armor key* sert de clé d'authentification. Ce *checksum* porte sur toute le message `AP_REQ` d'authentification du client[27] ;
- `enc-fast-req` qui contient une copie chiffrée avec l'*Armor key* du bloc `KDC-REQ-BODY` de la requête `TGS_REQ` du client. Le KDC doit préférer ce champ, car son chiffrement lui assure une protection en confidentialité et intégrité, contrairement à celui de la requête `TGS_REQ`.

Modification des messages `TGS_REP` La méthode de protection des réponses `TGS_REP` est identique à celle opérée sur les réponses `AS_REP` :

- la clé de renforcement est calculée en combinant la clé contenue dans le champ `strengthen-key` du bloc `KrbFastResponse`[28] et la clé de session $S_{C,K}$;

[27] Le message `AP_REQ` est contenu dans un bloc de pré-authentification de type `PA-TGS-REQ`.

[28] Bloc qui, pour rappel, est chiffré avec l'*Armor key*.

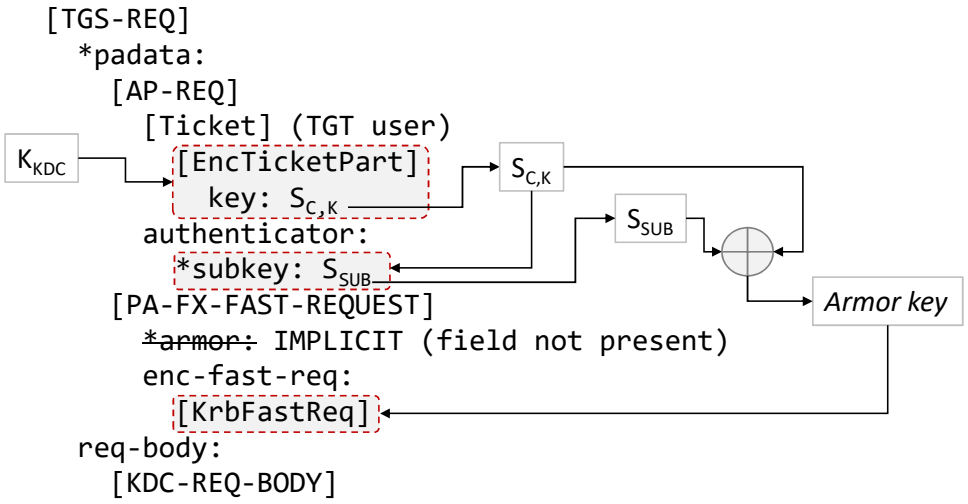

Fig. 7. Requête `TGS_REQ` protégée par FAST en *Implicit armor*

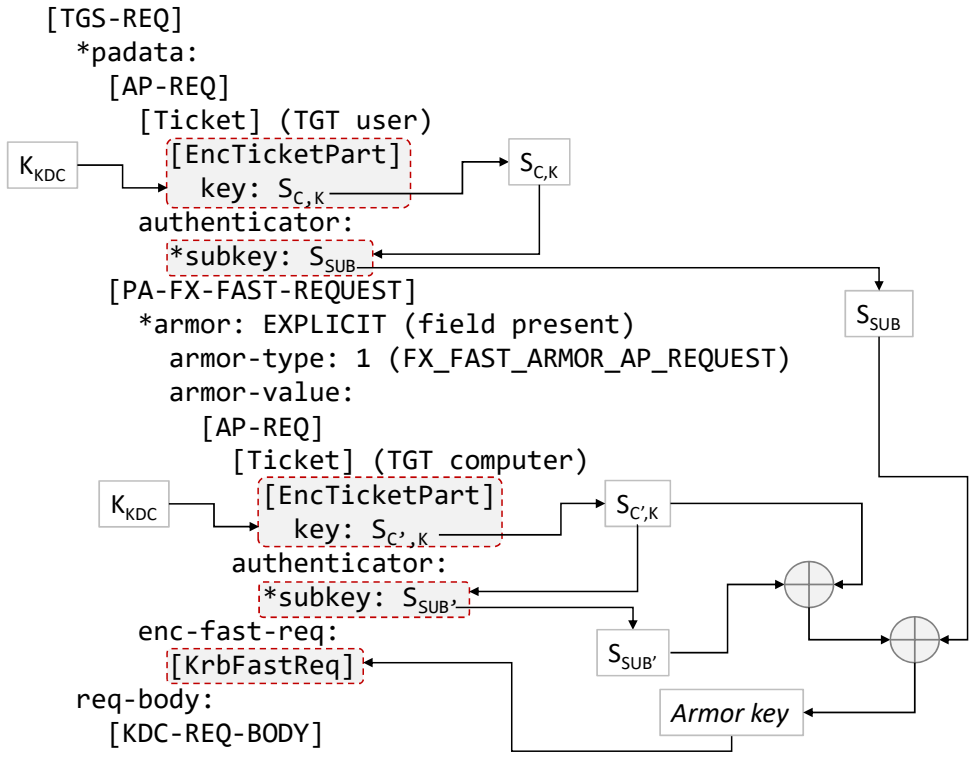

Fig. 8. Requête `TGS_REQ` protégée par FAST en *Explicit armor*

– cette clé de renforcement est utilisée pour protéger, en lieu et place de la clé $\texttt{S}_{\texttt{C,K}}$, la partie chiffrée de la réponse (champ `enc-part` contenant une structure `EncKDCRepPart`).

Configuration de FAST Pour que FAST soit mis en œuvre, il faut qu'il soit activé au niveau de la configuration des clients et des KDC.

Côté client, le paramètre[29] « *Kerberos client support for claims, compound authentication and Kerberos armoring* » (`EnableCbacAndArmor`) permet d'activer le support de FAST et des revendications. Celui-ci peut prendre les valeurs 0 ou 1. La valeur 1 stipule au client d'utiliser FAST pour les requêtes `AS_REQ` et `TGS_REQ` si le domaine le supporte (voir ci-dessous) et de demander que les revendications utilisateur soient incluses dans la PAC. La demande de revendication est caractérisée par la présence, dans les requêtes `AS_REQ` et `TGS_REQ` d'un bloc de type `PA-PAC-OPTIONS` [12] où le bit `Claims` est activé dans le champ `KerberosFlags`. Quant au paramètre « *Fail authentication requests when Kerberos armoring is not available* » (`ClientRequireFast`), il force le client à rejeter les réponses `AS_REP` et `TGS_REP` si celles-ci ne sont pas protégées par FAST.

Côté KDC, le paramètre « *KDC support for claims, compound authentication and Kerberos armoring* » (`CbacAndArmor`) permet d'activer le support de FAST et des revendications. Celui-ci peut prendre les valeurs suivantes :

– 0 (`NoCbacAndArmor`[30]) : le domaine ne supporte ni FAST ni les revendications ;
– 1 (`MixModeCbacAndArmor`) : paramètre recommandé lorsque tous les DC ne sont pas de type Windows Server 2012 ou supérieur. Dans ce mode, le domaine annonce (voir ci-dessous) qu'il supporte FAST et les revendications. Celles-ci seront incluses dans les données d'autorisation des tickets Kerberos que si les clients en font la demande ;
– 2 (`FullModeCbacAndArmor`) : paramètre utilisable uniquement lorsque tous les DC sont de type Windows Server 2012 ou supérieur. Dans ce mode, le domaine annonce qu'il supporte FAST. Quant aux revendications, elles sont systématiquement incluses dans les données d'autorisation des tickets Kerberos ;
– 3 (`FullModeCbacAndRequireArmor`) : paramètre utilisable seulement lorsque tous les DC sont de type Windows Server 2012 ou supérieur. Dans cette configuration, toutes les requêtes `AS_REQ` ou `TGS_REQ`

[29] Tous les paramètres Kerberos peuvent être gérés par GPO : *Computer Configuration*, *Administrative Templates*, *System*, *Kerberos* (partie cliente) ou *KDC* (partie KDC).
[30] Les noms des différents niveaux sont issus du modèle d'administration `kdc.admx`.

doivent être protégés par FAST et les revendications sont systématiquement incluses dans les données d'autorisation des tickets Kerberos.

L'attribut `msDS-SupportedEncryptionTypes` du compte `krbtgt` permet d'annoncer aux clients le support par le domaine de FAST ou les revendications. Sa valeur est mise à jour automatiquement par les contrôleurs de domaine en fonction du paramètre `CbacAndArmor` vu ci-dessus et ne doit pas être modifiée manuellement. Si les 16 bits de poids faible de l'attribut `msDS-SupportedEncryptionTypes` indiquent effectivement le support d'algorithmes de chiffrement (en particulier d'AES), les 16 bits de poids fort indiquent, quant à eux, le support de fonctionnalités particulières [14] :

- le bit `FAST-supported` spécifie que le service[31] supporte FAST ;
- le bit `Claims-supported` spécifie que le service (*i.e.* le domaine) supporte les revendications ;

Afin de transmettre la valeur de cet attribut aux clients, une structure `PA-SUPPORTED-ENCTYPES` [13] est ajoutée dans le champ `encrypted-pa-data`[32] de la partie `EncKDCRepPart` de la réponse du KDC. La valeur de l'entier est simplement reprise de l'attribut `msDS-SupportedEncryptionTypes` du compte `krbtgt`.

```
[EncKDCRepPart]
  ...
  *encrypted-pa-data:
    [PA-DATA]
      [1] padata-type: 167 (PA-PAC-OPTIONS), KerberosFlags: 0x80000000 (Claims)
      [2] padata-type: 165 (PA-SUPPORTED-ENCTYPES), EncryptionTypes: 0x5001f
          0x50000: Claims-supported, FAST-supported
          0x0001f: AES256, AES128, RC4-HMAC, DES-CBC-MD5, DES-CBC-CRC
```

9.2 Stratégie d'authentification

Comme vu dans la section 9.1 (paragraphe« Apports de FAST pour les échanges AS »), FAST permet de limiter le périmètre de diffusion des secrets d'authentification aux ordinateurs. Les stratégies d'authentification viennent en complément et permettent de restreindre les ordinateurs pouvant être utilisés pour l'authentification. Cela réduit encore le périmètre de diffusion des secrets d'authentification et améliore la protection des niveaux d'administration.

[31] Comprendre ici le domaine puisqu'il s'agit du compte `krbtgt`.

[32] Le champ `encrypted-pa-data` n'est pas présent dans la définition d'origine de la structure `EncKDCRepPart`. Il est défini dans la RFC 6806 [4].

Les stratégies d'authentification sont apparues avec Windows Server 2012 R2. Elles permettent, au niveau des contrôleurs de domaine, d'appliquer des restrictions lors des demandes d'authentification (Kerberos ou NetLogon pour NTLM). Elles sont définies par des objets de classe `msDS-AuthNPolicy` [6] devant être créés dans le conteneur `CN=AuthN Policies,CN=AuthN Policy Configuration,CN=Services` de la partition de configuration de la forêt. La configuration des stratégies d'authentification étant située dans la partition configuration de la forêt, elles sont donc appliquées à toute la forêt et modifiables, par défaut, que par les administrateurs de l'entreprise et aux administrateurs du domaine racine.

Les restrictions peuvent être appliquées à trois types de comptes, différenciés selon leur classe :

- aux comptes utilisateur (classe `User`) ;
- aux comptes de service, ce qui correspond aux comptes de service administrés (*Standalone Managed Service Accounts*, classe `ms-DS-Managed-Service-Account`) ou les comptes de service administrés de groupe (*Group Managed Service Accounts* , classe `ms-DS-Group-Managed-Service-Account`) ;
- aux comptes machine (classe `Computer`).

Plusieurs attributs de la classe `msDS-AuthNPolicy` définissent la stratégie d'authentification suivant le type de compte :

- `msDS-UserTGTLifetime`, `msDS-ServiceTGTLifetime`, ainsi que `msDS-ComputerTGTLifetime` : durée de vie des TGT ;
- `msDS-UserAllowedToAuthenticateFrom` et `msDS-ServiceAllowedToAuthenticateFrom` : critères sur le périphérique depuis lequel l'authentification du compte est réalisée (applicable uniquement pour les utilisateurs et les comptes de service) ;
- `msDS-UserAllowedToAuthenticateTo`, `msDS-ServiceAllowedToAuthenticateTo` et `msDS-ComputerAllowedToAuthenticateTo` : critères sur les comptes pouvant s'authentifier auprès d'un service s'exécutant sous l'identité d'un compte où la restriction s'applique.

Ces deux derniers attributs sont l'objet des paragraphes suivants.

Restrictions *AllowedToAuthenticateFrom* Les attributs de type *AllowedToAuthenticateFrom* permettent d'appliquer des restrictions sur le périphérique *(Device)* depuis lequel un compte est autorisé à s'authentifier. Avec Kerberos, la vérification a lieu lors des requêtes `AS_REQ`.

Pour pouvoir appliquer des restrictions sur le périphérique (*i.e.* la machine), il faut que le KDC dispose du TGT de celle-ci lorsqu'il reçoit les requêtes `AS_REQ`. Ceci nécessite donc qu'elles soient blindées avec un blindage de type *Explicit armor*[33].

Ces restrictions peuvent être appliquées aux comptes utilisateur et de service, mais pas aux comptes machine. Cela n'a évidement pas de sens d'appliquer des restrictions de type *From* aux comptes machine puisqu'il s'agit de restrictions portant sur elles-mêmes et que, pour les mêmes raisons, elles ne peuvent blinder leur requête `AS_REQ`.

Les restrictions prennent la forme d'un descripteur de sécurité qui est confronté aux données d'autorisation contenues dans le TGT de la machine. Le descripteur permet de définir des ACE portant sur les identifiants de sécurité (SID) ou sur les revendications Utilisateur *(User Claims)* de la machine.

Dans l'exemple ci-dessous, un TGT ne sera délivré à un utilisateur que si celui-ci effectue sa demande depuis une machine étant membre (`Member of any`) du groupe `DOM\Admin_Wks` et (`And`) ayant une revendication utilisateur (`User.`) appelée « description » et valant (`Any`) « adm » :

```
(Member of any({Test (DOM\Admin_Wks)}) And (User.description Any of {"adm"}))
```

Notes : attention, les restrictions *AllowedToAuthenticateFrom* concernent la machine depuis laquelle l'authentification est réalisée. Le terme « Utilisateur » fait donc référence à celle-ci : les SID ou les revendications utilisateur référencés dans le descripteur de sécurité sont ceux associés au compte de la machine dans le domaine. De plus, l'exemple ci-dessus est issu de l'interface graphique qui affiche les descripteurs de sécurité sous forme « compréhensible ». La commande PowerShell `Get-ADAuthenticationPolicy` permet d'afficher les SDDL des descripteurs de sécurité sous leur forme « brute »[34], ce qui donne :

```
(XA;OICI;CR;;;WD;(
  (Member_of_any{SID(<SID du groupe DOM\Admin_Wks>)})
    &&
  (@USER.ad://ext/description:88d46537a8db15a9 Any_of {"adm"})
))
```

[33] Pour rappel, comme vu dans la section 9.1 (paragraphe « Modification des messages `AS_REQ` »), si FAST et mis en œuvre, le blindage de type *Explicit armor* est obligatoire pour les requêtes `AS_REQ`.

[34] Pour ne pas dire brutale...

Restrictions *AllowedToAuthenticateTo* Les attributs de type *AllowedToAuthenticateTo* permettent d'appliquer des restrictions sur les comptes autorisés à s'authentifier auprès d'un service s'exécutant sous l'identité du compte auquel la stratégie est appliquée[35]. Avec Kerberos, la vérification est réalisée lors des demandes de ticket de service (requêtes `TGS_REQ`).

Ces restrictions peuvent être appliquées à n'importe quel type de compte (utilisateur, service, machine), ces trois types de compte pouvant être utilisés comme entité de sécurité sous lequel un processus peut s'exécuter.

Les restrictions prennent la forme d'un descripteur de sécurité autorisant des entités de sécurité. Les critères d'autorisation peuvent être des SID ou des revendications utilisateur. Il est également possible de spécifier des critères sur le périphérique de l'utilisateur (SID ou revendications périphérique) si l'authentification composée est prise en charge (voir section 9.4). Les restrictions d'authentification sont appliquées lorsqu'un utilisateur demande un ticket de service et qu'une stratégie d'authentification est appliquée au compte associé. Dans ce cas, les données d'autorisation contenues dans le TGT (partie `AP_REQ`) présenté par l'utilisateur sont confrontées au descripteur de sécurité spécifié par la stratégie.

Dans l'exemple ci-dessous, un ticket de service ne sera délivré à un utilisateur que si celui-ci possède une revendication utilisateur appelée `AuthenticationSilo` et valant `ROUGE` et s'il est membre du groupe « Administrateurs du domaine ».

```
((User.AuthenticationSilo Equals "ROUGE")
  And
Member of each({Domain Admins (DOM\Domain Admins)}))
```

Ce qui donne le SDDL suivant :

```
(XA;OICI;CR;;;WD;(
  (@USER.ad://ext/AuthenticationSilo == "ROUGE")
    &&
  (Member_of {SID(DA)})
))
```

Affectation des stratégies d'authentification L'affectation des stratégies d'authentification ne se fait pas au niveau de la stratégie mais au niveau des objets qu'elles impactent. Ainsi, pour chacun de trois types

[35] La jointure service/compte est effectuée par l'attribut *Service Principal Name* du compte.

de compte (utilisateur, service et ordinateur), la stratégie appliquée au compte est déterminée par l'attribut `msDS-AssignedAuthNPolicy`. Celui-ci référence le *Distinguish name* de la stratégie devant s'appliquer. Afin de déterminer rapidement les *principals* affectés à une stratégie donnée, il existe l'attribut `msDS-AssignedAuthNPolicyBL` qui est l'attribut *Backlink* de `msDS-AssignedAuthNPolicy`.

Note : l'affection étant au niveau du compte, il est donc impossible d'utiliser des groupes pour affecter les stratégies d'authentification à un ensemble de comptes.

La gestion des stratégies d'authentification peut être effectuée avec l'*Active Directory Administrative Center* ou par PowerShell et les cmdlets `*-ADAuthenticationPolicy`.

Enfin, afin d'éviter un blocage complet de l'authentification, le compte intégré Administrateur du domaine (RID 500) est totalement exempté : même si une stratégie est appliquée (directement ou par un silo) à ce compte, celle-ci n'est pas effective. Ceci permet, en ultime recours, d'utiliser le compte intégré Administrateur pour corriger une éventuelle erreur de configuration. Ceci renforce le rôle « bris de glace » de ce compte et le fait qu'il ne doit jamais être utilisé au quotidien.

9.3 Silo d'authentification

Les silos d'authentification sont apparus simultanément avec les stratégies d'authentification (Windows Server 2012 R2). Leur objectif est de faciliter la mise en place de restriction d'authentification en combinant revendications et stratégies d'authentification.

Ils sont définis par des objets de classe `msDS-AuthNPolicySilo` [7] situés dans le conteneur `CN=AuthN Silos,CN=AuthN Policy Configuration,CN=Services` de la partition de configuration de la forêt. Comme pour les stratégies d'authentification, les silos sont appliqués à toute la forêt et ne peuvent, par défaut, être modifiés que par les administrateurs de l'entreprise et les administrateurs du domaine racine.

Un silo est caractérisé par :

- un nom ;
- une liste de membres (`msDS-AuthNPolicySiloMembers`) pouvant appartenir au silo *(Permitted Accounts)*. Il peut s'agir de comptes utilisateur, de service ou machine ;
- une stratégie d'authentification qui est appliquée (et de manière préférentielle) à tous les membres du silo. En réalité, il y a trois attributs définisant la stratégie d'authentification (`msDS-UserAuthNPolicy`,

`msDS-ServiceAuthNPolicy` et `msDS-ComputerAuthNPolicy`), ce qui permet de différencier celle appliquée suivant le type de compte.

Pour qu'un compte (utilisateur, service, machine) soit considéré comme « membre d'un silo », il faut qu'il soit autorisé au niveau du silo (attribut `msDS-AuthNPolicySiloMembers` au niveau du silo) et qu'il se déclare comme appartenant au silo (attribut `msDS-AssignedAuthNPolicySilo` au niveau du compte). Ce double mode d'affectation permet de s'assurer qu'un administrateur d'un compte donné ne puisse pas affecter celui-ci à un silo sans qu'un administrateur du silo ne l'ait autorisé au niveau du silo.

Deux attributs *Backlink* sont également disponibles :

- au niveau des comptes, `msDS-AuthNPolicySiloMembersBL` (*Backlink* de `msDS-AuthNPolicySiloMembers`) qui permet de voir les silos qui peuvent être affectés au compte ;
- au niveau du silo, `msDS-AssignedAuthNPolicySiloBL` (*Backlink* de `msDS-AssignedAuthNPolicySilo`) qui permet de lister les comptes étant affectés au silo.

Comme pour les stratégies d'authentification, les affectations sont effectuées directement au niveau des comptes ce qui ne permet pas d'utiliser des groupes pour l'affectation.

L'intérêt d'un silo est double :

- d'une part, les membres d'un silo se voient automatiquement attribuer une revendication utilisateur appelée `AuthenticationSilo` et valant le nom du silo. L'objectif de cette revendication est d'être utilisée dans des *Conditional ACEs* pour vérifier l'appartenance d'un utilisateur ou d'une machine à un silo donné ;
- si un compte est membre d'un silo et qu'une stratégie d'authentification est appliquée au niveau du silo, cette dernière est prioritaire à une éventuelle stratégie qui pourrait être affectée au niveau du compte.

Les silos sont particulièrement adaptés pour restreindre un groupe d'utilisateurs à s'authentifier uniquement depuis un groupe de machines. Pour cela, il suffit de créer un silo avec les caractéristiques suivantes :

- membres :
  - comptes utilisateur pour lesquels la restriction doit s'appliquer,
  - comptes machine depuis lesquels les utilisateurs membres seront autorisés à s'authentifier ;

– stratégie d'authentification : restriction l'authentification des utilisateurs aux machines membres du silo. Ceci peut aisément être mis en place en imposant que l'ordinateur, depuis lequel les utilisateurs membres s'authentifient possède une « revendication utilisateur » appelée « silo » et valant le nom du silo ;

Il suffit ensuite d'associer et d'appliquer la stratégie aux membres du silo.

Dans le modèle d'administration décrit dans la section 2, les silos viennent compléter le mécanisme des *Protected users* et du blindage Kerberos en contrôlant les systèmes depuis lesquels les administrateurs de niveau ROUGE peuvent s'authentifier. Dans l'idéal, il doit s'agir des contrôleurs de domaine et des postes d'administration de niveau ROUGE.

9.4 L'authentification composée

Principe L'authentification composée *(Compound Identity)* est l'aboutissement de toutes les technologies vues précédemment. Le principe consiste à inclure dans les données d'autorisation d'un utilisateur les données d'autorisation de la machine depuis laquelle il opère. Ces données du périphérique (« *Device* ») peuvent alors être référencées dans les *Conditional ACEs* afin d'appliquer des restrictions sur l'utilisateur mais également sur sa machine.

L'authentification composée a lieu lors des requêtes `TGS_REQ` d'un compte utilisateur ou de service[36]. Si le compte souhaite composer son authentification, il doit blinder sa demande avec un *Explicit armor*. Cela permet au KDC de disposer du TGT de la machine et donc des données d'autorisation de celle-ci. Lors de la génération des données d'autorisation de l'utilisateur, en plus de ses propres données (SID du domaine, RID des groupes, extra SID, revendications utilisateur), les données d'autorisation de la machine sont également incluses. Cela se traduit dans la PAC de l'utilisateur par :

– un bloc de type `PAC_DEVICE_CLAIMS_INFO` qui contient les revendications utilisateur de la machine qui sont maintenant les revendications périphérique *(Device claims)* de l'utilisateur ;
– un bloc de type `PAC_DEVICE_INFO` qui contient les données d'autorisation de la machine (AccountDomainId, UserId, PrimaryGroupId, AccountGroupIds, etc.) ;

[36] Les comptes machine ne peuvent pas en bénéficier car ils ne peuvent blinder leurs requêtes Kerberos.

– l'ajout du SID `COMPOUNDED_AUTHENTICATION`[37] dans les ExtraSids de l'utilisateur.

Comme pour les revendications utilisateur, lorsqu'un système Windows reçoit un ticket avec une PAC contenant des données d'autorisation issues de son périphérique, il les inclut dans le *Token* de l'utilisateur au moment de sa création. Ainsi :

– les données d'autorisation du périphérique contenues dans le bloc `PAC_DEVICE_INFO` sont mises dans le champ `_TOKEN.pClaimAttributes.pDeviceGroups` du *token* ;
– les revendications périphérique, contenues dans le bloc `PAC_DEVICE_CLAIMS_INFO`, sont mises dans la partie `DeviceClaims` du *token* (`_TOKEN.pClaimAttributes.pDeviceSecurityAttributes`[38]).

Process Hacker permet d'afficher les revendications périphérique d'un *Token* dans l'onglet « *Claims* » de la page des propriétés avancées d'un *Token*, sous la dénomination « *Device claims* » (voir figure 9).

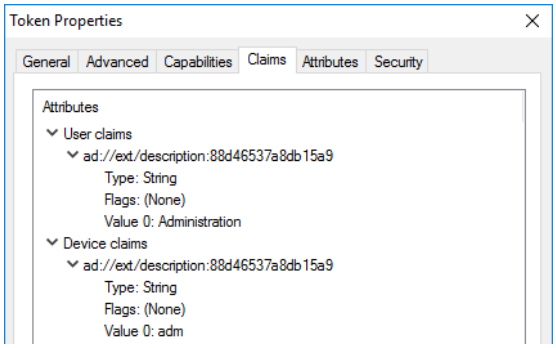

Fig. 9. Affichage des revendications avec Process Hacker

Filtrage Étant désormais situées dans le *token* de l'utilisateur, les données d'autorisation des périphériques peuvent être utilisées dans le contrôle d'accès *via* les *Conditional ACEs*. Cela permet de restreindre les accès à des ressources uniquement depuis des ordinateurs identifiés.

Les directives `Device_Member_of` ou `Device_Member_of_Any`[39] permettent de vérifier la présence de SID dans le champ `pDeviceGroups`.

[37] S-1-5-21-0-0-0-496
[38] Structure de type `AUTHZBASEP_SECURITY_ATTRIBUTES_INFORMATION`.
[39] Il est également possible d'utiliser les directives avec la logique inverse : `Not_Device_Member_of` ou `Not_Device_Member_of_Any`.

Par exemple, l'ACE ci-dessous permet d'autoriser (`XA`) la lecture d'un fichier (`0x1200a9`) à tous les utilisateurs (`WD`) mais uniquement depuis un contrôleur de domaine (`DD`).

```
(XA;;0x1200a9;;;WD;(Device_Member_of_any {SID(DD)}))
```

Quant à la directive `@DEVICE`, elle permet de spécifier des critères sur les revendications périphérique. L'ACE ci-dessous n'autorise l'accès que si une revendication périphérique issue de l'AD (`@DEVICE.ad`) appelée « `AuthenticationSilo` » est présente et vaut la valeur « `ROUGE` ».

```
(XA;;0x1200a9;;;WD;(@DEVICE.ad://ext/AuthenticationSilo Any_of {"ROUGE"}))
```

Enfin, il est possible de confronter les revendications entre elles. L'exemple ci-dessous n'autorise l'accès que si le silo de l'utilisateur (`@USER.ad:AuthenticationSilo`) est identique à celui de son ordinateur (`@DEVICE.ad:AuthenticationSilo`) et si le niveau d'habilitation de l'utilisateur (`@USER.ad:AuthorizationLevel`) est supérieur ou égal au niveau de classification de la ressource (`@RESOURCE.Confidentiality_MS`).

```
D:PAI(XA;;FX;;;WD;(
  (@USER.ad://ext/AuthenticationSilo == @DEVICE.ad://ext/AuthenticationSilo)
    &&
  (@USER.ad://ext/AuthorizationLevel >= @RESOURCE.Confidentiality_MS)
))
```

Note : l'authentification composée n'est applicable que pour les accès distants. Pour les accès locaux, il est possible d'utiliser le SID `INTERACTIVE` (`IU`) pour autoriser les utilisateurs authentifiés interactivement. L'exemple suivant autorise les utilisateurs ayant une revendication périphérique ou étant authentifiés interactivement :

```
D:(XA;OICI;0x1200a9;;;BU;(
  (@DEVICE.ad://ext/description:88d46537a8db15a9 Any_of {"adm"})
    ||
  (Member_of {SID(IU)})
))
```

Configuration et activation Pour que l'authentification composée soit mise en œuvre, il faut que le client la demande auprès du KDC. Cette demande prend la forme d'une requête `TGS_REQ` avec un blindage de type

Explicit armor. Un ensemble de paramètres régissent le comportement du client et du KDC.

Tout d'abord, le support des revendications, de l'authentification composée et du blindage Kerberos doit être activé côté KDC et côté client (voir section 9.2, paragraphe « Configuration de FAST »).

Dans le fonctionnement par défaut, le client commence par effectuer une requête `TGS_REQ` « normale », c'est-à-dire avec un blindage de type *Implicit armor* (voir section 9.1, paragraphe « Modification des messages `TGS_REQ` »). Avec ce type de blindage, le KDC ne dispose pas du TGT de la machine et donc des données d'autorisation de celle-ci. Il est donc dans l'incapacité de réaliser l'authentification composée. Le ticket de service retourné au client n'est donc pas « composé ». Cependant, dans la réponse (message `TGS_REP`), le KDC indique au client les algorithmes supportés par le service demandé (cette information prend la forme d'un entier de 32 bits contenu dans un bloc de type `PA-SUPPORTED-ENCTYPES`). La valeur de l'entier est simplement reprise de l'attribut `msDS-SupportedEncryptionTypes` du compte associé au SPN (*Service Principal Name*) du service demandé.

Comme vu dans précédemment cet attribut est mis à jour automatiquement par les services afin d'indiquer le support de fonctionnalités particulières (16 bits de poids forts). Ainsi le bit `Compound-identity-supported` spécifie que le service supporte l'authentification composée.

En analysant la réponse et le bloc `PA-SUPPORTED-ENCTYPES`, si le bit `Compound-identity-supported` est positionné, le client détecte que le service supporte la composition alors qu'il ne dispose pas d'un ticket de service composé. Une nouvelle demande de ticket est effectuée mais avec, cette fois-ci, un blindage de type *Explicit armor* qui permet de transmettre le TGT de la machine de l'utilisateur. Le KDC peut donc procéder à la composition. Ce mode de fonctionnement permet de demander l'authentification composée uniquement lorsque le service le supporte.

Ainsi, pour activer l'authentification composée pour un service donné, il faut que le bit `Compound-identity-supported` soit positionné dans l'attribut `msDS-SupportedEncryptionTypes` du compte associé au service. Si ce bit peut être activé manuellement pour les comptes utilisateur ou les comptes de service, le fonctionnement est plus subtil pour les comptes machine. En effet, à l'image du compte `krbtgt` (voir section 9.2, paragraphe « Configuration de FAST », cet attribut est mis à jour automatiquement par les ordinateurs en fonction de leur caractéristique[40].

[40] Cet attribut est autorisé en écriture pour le principal du compte associé.

Un système Windows positionne le bit `Compound-identity-supported` en fonction du paramètre « Support Compound authentication »[41] (`CompoundIdEnabled`). Celui-ci peut prendre trois valeurs :

- « Jamais » : doit être choisi quand aucun service supportant l'authentification composée n'est associé au compte de la machine. Dans ce cas, le système ne positionne donc pas le bit ;
- « Automatique » : le système positionne le bit si au moins un service est inscrit comme supportant l'authentification composée. Un tel service doit s'inscrire en créant une clé `HKLM\SYSTEM\CurrentControlSet\Services\Netlogon\Parameters\CompoundIdentity`. Dans la pratique, ceci est rarement utilisé ;
- « Toujours » : le système positionne systématiquement le bit. C'est cette méthode qui est privilégiée *via* le déploiement du paramètre avec une GPO appliquée aux systèmes nécessitant de recevoir des authentifications composées.

Deux autres paramètres de configuration peuvent aussi être utilisés :

- Partie KDC : « *Request compound authentication* » (RequestCompoundId) qui permet simuler l'activation du bit `Compound-identity-supported` pour tous les comptes ;
- Partie Kerberos : « *Always send compound authentication first* » (AlwaysSendCompoundId) qui permet de forcer le client à blinder ses requêtes `TGS_REQ` avec un blindage de type *Explicit armor* sans prendre en compte la valeur de l'attribut `msDS-SupportedEncryptionTypes`.

10 Conclusion

La protection des secrets d'authentification doit être une priorité afin d'assurer la sécurité des comptes d'administration. Les mécanismes abordés dans cet article, en particulier les silos d'authentification, permettent la mise en place de protections avancées et efficaces.

Cependant, tout ceci n'a d'intérêt que si les domaines ont déjà une certaine « hygiène d'administration » : avoir peu de comptes d'administration de niveau ROUGE, utiliser des systèmes d'exploitation récents, être mis à jour régulièrement, avoir des sauvegardes protégées, etc.

[41] Et plus précisément son unique option de configuration : « *Support authorization with client device information* ».

Il ne faut surtout pas oublier que toutes les protections basées sur la composition de l'authentification ne sont d'aucune utilité si un attaquant dispose des secrets du compte *krbtgt* ou des comptes machine.

Quant aux revendications, elles étendent le modèle de sécurité et permettent une gestion plus fine des droits d'accès. Cependant, on peut être sceptique quant à leur mise en œuvre quand on sait déjà que les administrateurs ont du mal à gérer les droits d'accès basés sur les SID. De plus, ceci complexifie grandement l'analyse des permissions d'accès puisqu'il faut désormais prendre en compte les valeurs attributs dans l'Active Directory.

Remerciements Merci à tous les relecteurs pour leur amicale et très précieuse relecture.

Références

1. Aurélien Bordes. Kerberos. `http://aurelien26.free.fr/kerberos/`.
2. Aurélien Bordes. Secrets d'authentification épisode II : Kerberos contre-attaque. `https://www.sstic.org/2014/presentation/secrets_dauthentification_pisode_ii__kerberos_contre-attaque/`.
3. IETF. A Generalized Framework for Kerberos Pre-Authentication. `https://tools.ietf.org/html/rfc6113`.
4. IETF. Kerberos Principal Name Canonicalization and Cross-Realm Referrals. `https://tools.ietf.org/html/rfc6806`.
5. Microsoft. CLAIM_SECURITY_ATTRIBUTE_V1 structure. `https://msdn.microsoft.com/en-us/library/windows/desktop/hh448489.aspx`.
6. Microsoft. [MS-ADSC] : Active Directory Schema Classes, Class msDS-AuthNPolicy. `https://msdn.microsoft.com/en-us/library/dn410862.aspx`.
7. Microsoft. [MS-ADSC] : Active Directory Schema Classes, Class msDS-AuthNPolicySilo. `https://msdn.microsoft.com/en-us/library/dn409948.aspx`.
8. Microsoft. [MS-ADTS] : Active Directory Technical Specification, ValidateClaimDefinition. `https://msdn.microsoft.com/en-us/library/jj216791.aspx`.
9. Microsoft. [MS-DTYP] : Windows Data Types, CLAIM_SECURITY_ATTRIBUTE_RELATIVE_V1. `https://msdn.microsoft.com/en-us/library/hh877833.aspx`.
10. Microsoft. [MS-DTYP] : Windows Data Types, Conditional ACEs. `https://msdn.microsoft.com/en-us/library/hh877827.aspx`.
11. Microsoft. [MS-DTYP] : Windows Data Types, SYSTEM_RESOURCE_ATTRIBUTE_ACE. `https://msdn.microsoft.com/en-us/library/hh877837.aspx`.
12. Microsoft. [MS-KILE] : Kerberos Protocol Extensions, PA-PAC-OPTIONS. `https://msdn.microsoft.com/en-us/library/hh553950.aspx`.
13. Microsoft. [MS-KILE] : Kerberos Protocol Extensions, PA-SUPPORTED-ENCTYPES. `https://msdn.microsoft.com/en-us/library/ee896760.aspx`.
14. Microsoft. [MS-KILE] : Kerberos Protocol Extensions, Supported Encryption Types Bit Flags. `https://msdn.microsoft.com/en-us/library/ee808210.aspx`.

15. Microsoft. MSDN, AddResourceAttributeAce function. https://msdn.microsoft.com/en-us/library/windows/desktop/hh448447.aspx.
16. Microsoft. MSDN, Security Descriptor Definition Language for Conditional ACEs. https://msdn.microsoft.com/en-us/library/windows/desktop/dd981030.aspx.