Chemins de contrôle en environnement Active Directory

Lucas Bouillot et Emmanuel Gras
prenom.nom@ssi.gouv.fr

Agence Nationale de la Sécurité des Systèmes d'Information
Bureau Audits et Inspections

Résumé Cet article présente une méthode d'analyse de la sécurité des environnements Active Directory fondée sur l'établissement de relations entre les différents éléments composant un domaine et traduisant la maîtrise d'un objet sur un autre. Ces relations sont issues de sources multiples : analyse des permissions, appartenances aux groupes de sécurité, propriétés et hiérarchie des objets de l'annuaire, fichiers de GPO, mais aussi propriétés liées aux machines locales. La finalité de cette analyse est d'agréger ces relations sous forme de graphes afin de mettre en évidence des chemins de contrôle mettant en jeu des enchaînements non triviaux de relations et d'objets. En fournissant des outils permettant de mieux appréhender un domaine complexe, cette méthode peut servir à vérifier la bonne isolation d'un groupe d'administration du reste du domaine ou à mesurer l'étendue effective du pouvoir d'un compte. Toutes les étapes de notre méthode seront abordées : définition d'un ensemble de relations de contrôle, méthodes de relevés possibles, représentation et agrégation dans une base de données orientée graphe, puis exploitation et interprétation au travers de scénarios d'analyse.

1 Introduction

Active Directory est une solution déployée par un grand nombre d'entreprises pour la gestion et l'administration de leurs ressources informatiques comprenant un grand nombre de systèmes Windows. Du fait de son rôle central au sein du système d'information (SI), la compromission du domaine conduit généralement à un accès complet à toutes les ressources de l'entreprise. Une analyse des différentes possibilités de compromission d'une architecture AD est donc obligatoire, aussi bien dans un contexte d'audit en vue de durcissement, que lors d'une activité de réponse à incident.

Les analyses de sécurité effectuées aujourd'hui sont souvent centrées autour de deux axes : l'hygiène des comptes du domaine (en particulier des groupes d'administration), et l'audit des permissions sur les objets de l'annuaire Active Directory [6]. Si ces analyses sont assurément indispensables, la complexité des différents services offerts par Active Directory

(gestion centrale des utilisateurs et des machines, contrôle d'accès centralisé, déploiement et maintien d'une politique logicielle homogène, etc.) nous impose d'étendre notre étude de deux manières. En premier lieu, il est nécessaire de prendre en compte un ensemble d'objets plus large que ceux contenus dans l'annuaire uniquement : tous les éléments manipulés par Active Directory doivent être considérés, car parmi eux nombreux sont ceux qui peuvent avoir une importance du point de vue de la sécurité. En second lieu, il convient d'inspecter d'autres propriétés que les permissions positionnées par les ACE (*Access Control Entries*, qui constituent le cœur du contrôle d'accès discrétionnaire des systèmes Windows). En effet, plusieurs propriétés n'en faisant pas partie peuvent s'avérer d'importance, par exemple celle liant un compte du domaine aux machines sur lesquelles il se connecte.

Notre article se propose donc de décrire une méthode d'analyse des environnements Active Directory centrée sur des **chemins de contrôle** entre les objets du domaine. Chaque chemin est composé d'un ensemble de ce que nous appelons **relations de contrôle direct**, où chaque relation traduit la maîtrise d'un objet sur un autre au travers d'une propriété particulière. Cette méthode consiste dans un premier temps à définir et à relever ces relations de contrôle direct, en intégrant un maximum d'éléments AD, puis, dans un second temps, à les agréger afin de pouvoir générer les chemins de contrôle. L'agrégation d'un maximum de relations joue un rôle crucial pour l'intérêt de notre méthode, puisque c'est cela qui permettra d'obtenir des chemins bien plus longs et complexes que ceux révélés communément, jusque-là souvent basés uniquement sur les permissions de l'annuaire et l'appartenance aux groupes de sécurité.

La finalité d'une telle méthode est de pouvoir identifier et visualiser (notamment sous forme de graphes) les différents parcours possibles jusqu'à des cibles privilégiées du domaine (groupes d'administration, utilisateurs VIP, etc.). Ces parcours représentent les moyens pour un attaquant d'atteindre ces cibles, et pourront servir à identifier des déviances dans la gestion du domaine, à valider l'application d'un périmètre de sécurité autour des cibles considérées, mais également à révéler des moyens de persistance laissés par un attaquant après une compromission.

Dans cet article, nous présenterons dans un premier temps les différentes relations de contrôle direct identifiées, en explicitant les mécanismes sur lesquels elles reposent, ainsi que les moyens de les relever. Puis, dans un second temps, nous détaillerons les moyens techniques retenus pour l'implémentation de nos outils, notamment pour le stockage, le calcul, et la visualisation des chemins de contrôle. Enfin nous décrirons des scénarios

concrets d'analyse, mettant en évidence des déviances, ou des possibles traces de compromission du domaine.

2 Active Directory, concepts et rappels

Cette partie a pour but de présenter quelques notions propres à Active Directory utilisées dans la suite de l'article. Les descriptions ne sont pas exhaustives, mais se concentrent sur les éléments importants dans le cadre du relevé et de l'analyse des relations de contrôle.

2.1 Architecture générale

Active Directory est une solution Microsoft de gestion de systèmes d'information, qui permet notamment de mettre en place :

- un **annuaire de ressources**, basé sur LDAP ;
- un **mécanisme d'authentification centralisé**, basé sur Kerberos ;
- une **politique logicielle homogène**, basée entre autres sur SMB.

Son architecture repose sur les notions de **domaine** et de **forêt**, ainsi que sur des serveurs centraux appelés **contrôleurs de domaine** (DC) regroupant l'ensemble des services et données.

Une forêt est une instance Active Directory, regroupant une infrastructure logique commune (données de schéma et de configuration), ainsi qu'un ou plusieurs domaines, qui contiennent eux une collection d'objets hiérarchisés représentant des ressources du SI et partageant des politiques d'administration, de sécurité, etc. Des domaines peuvent être liés entre eux par des relations d'approbation, permettant d'autoriser l'accès à des ressources locales à des membres d'un domaine distant (pouvant appartenir à une forêt distante). Ces relations, pouvant être unidirectionelles ou bidirectionnelles, filtrées ou non, transitives ou non, sont implicites entre les domaines d'une même forêt, et placent ainsi la limite de sécurité Active Directory au niveau de la forêt et non du domaine.

Les contrôleurs de domaine eux sont propres à un domaine. Ils présentent les différents services nécessaires au bon fonctionnement (LDAP, SMB, DNS, Netlogon, Kerberos, etc.), réalisent l'authentification des utilisateurs et des machines, et hébergent chacun une instance de la base de données du domaine `ntds.dit` utilisant la technologie Microsoft *Extensible Storage Engine* (ESE ou Jetblue). Des flux de réplication entre contrôleurs permettent de s'assurer de la propagation des modifications locales à chaque DC.

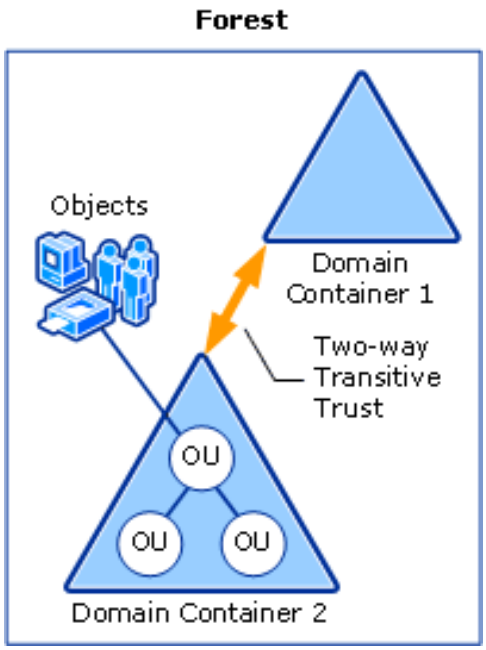


Figure 1. Forêt Active Directory comportant deux domaines (source : [22]).

2.2 Objets de l'annuaire

L'annuaire Active Directory est un annuaire accessible au travers du protocole LDAP, contenant des objets hiérarchisés. Il expose premièrement une racine accessible sans authentification appelée *RootDSE* qui fournit un ensemble d'informations sur le domaine et le contrôleur actuels. Puis, il segmente les données dans des partitions appelées *Naming Context* (NC). On peut ainsi retrouver [1] :

- au niveau de la forêt, les partitions `Configuration`, `Schema` ou `ForestDnsZones`, qui contiennent respectivement :
  - les objets de configuration pour la forêt, par exemple les sites, services et objets d'extension du contrôle d'accès,
  - la définition de toutes les classes d'objets et leurs attributs (qu'il est possible d'étendre en rajoutant de nouvelles entrées),
  - les informations de zone DNS pour la forêt ;
- au niveau du domaine, les partitions de domaine et `DomainDnsZones`, qui contiennent respectivement :
  - tous les objets propres au domaine en question, à savoir les utilisateurs, machines, groupes, OU, etc.,
  - les informations de zone DNS propres au domaine.

1. Les partitions de zone DNS peuvent ne pas être présentes si l'instance Active Directory utilise un service DNS externe.

L'organisation des objets est ensuite réalisée à travers différents conteneurs, et notamment les *Organizational Unit* (OU), sur lesquels s'appliquent les stratégies logicielles et les délégations de droits. Chaque objet est désigné de manière unique par son *Distinguished Name* (DN), représentant le chemin complet depuis la racine du domaine jusqu'à lui (par exemple : `CN=JohnDoe,OU=Corsaires,OU=Utilisateurs,DC=domaine`). Enfin, chaque objet, peu importe sa partition, est un *securable object* et possède donc un descripteur de sécurité permettant d'en contrôler l'accès (cf. partie 2.4).

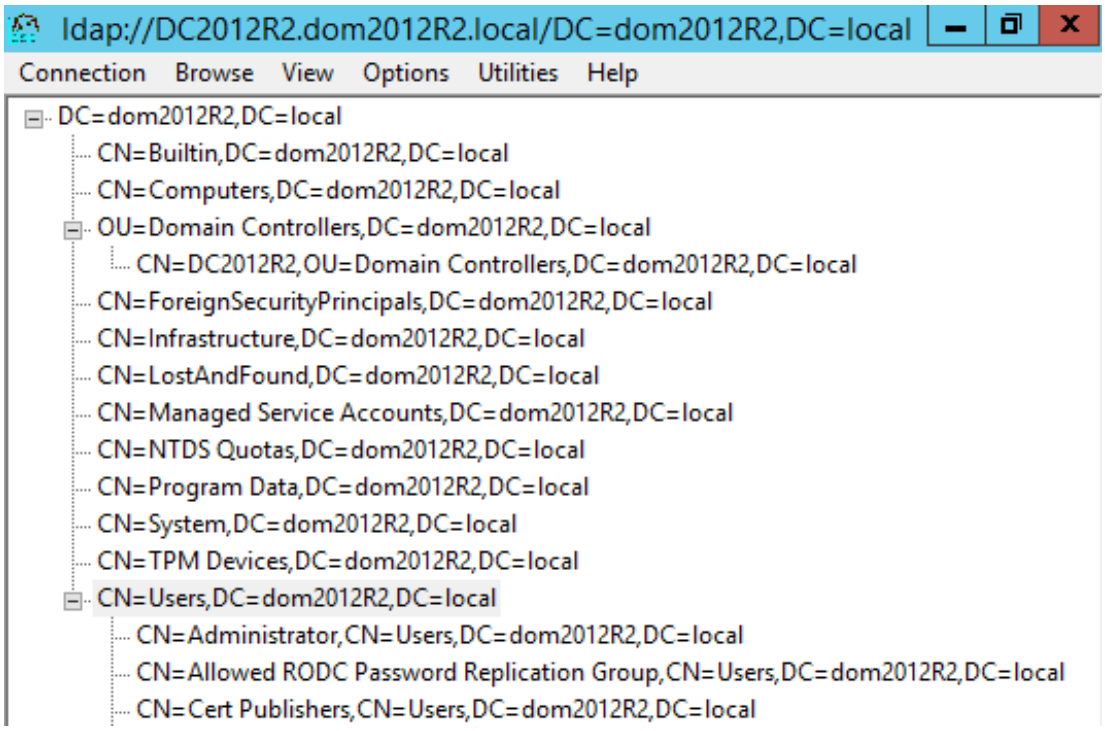


Figure 2. Vue de la hiérarchie d'objets d'une partition de domaine `dom2012R2.local` d'un annuaire LDAP Active Directory avec l'outil `LDP`.

Concernant leur définition, tous les objets sont des instances d'un ou plusieurs types d'objet spécifiés par leur attribut `objectClass`. Chaque classe d'objet :

- est construite à partir d'une hiérarchie d'une ou plusieurs classes mères ;
- hérite de ces classes mères un ensemble d'attributs ;
- définit d'autres attributs qui lui sont propres, facultatifs ou obligatoires.

Par exemple, une machine du domaine est représentée par une instance de la classe `Computer`, dont le chemin d'héritage est `Computer` ← `User` ← `Organizational-Person` ← `Person` ← `Top`. Elle définit elle-même l'attribut `Operating-System`, mais hérite par exemple des attributs `Unicode-Pwd` de la classe `User` et `NT-Security-Descriptor` de la classe `Top`. Au sens Active Directory, une machine du domaine est donc un type particulier d'utilisateur. Il faudra donc considérer cet héritage lors de notre analyse.

Toutes les classes et attributs sont eux-mêmes des objets de l'annuaire, présents dans la partition schéma, du type `Attribute-Schema` et `Class-Schema`. L'annuaire et les objets qu'il contient sont ainsi extensibles arbitrairement par des applications tierces, puisqu'il est possible de rajouter de nouvelles classes d'objets et de nouveaux attributs simplement en créant de nouvelles entrées dans le schéma.

2.3 Stratégies de groupes

Les stratégies de groupes (ou GPO - *Group Policy Objects*) sont des objets permettant de garder un parc de machines et un environnement utilisateur homogènes, *via* la définition et l'application d'un ensemble de paramètres de configuration pour les systèmes Windows. Ces objets sont liés à des conteneurs dans l'annuaire et permettent de définir des paramètres de configuration pour les objets contenus, d'exécuter des scripts, de déployer des logiciels, etc.

Une GPO, représentée par un GUID, est composée de deux parties :

- un **objet de GPO** appelé GPC - *Group Policy Container*, de classe `Group-Policy-Container`, enregistré dans l'annuaire (DN : `CN={<GUID>},CN=Policies,CN=System,DC=<domaine>`) regroupant les métadonnées de la GPO, et dont le DN est lié aux conteneurs auxquels la GPO doit s'appliquer ;
- un **dossier de GPO**, enregistré dans le partage réseau Sysvol, distribué entre les DC (chemin réseau : `\\<domaine>\SYSVOL\<domaine>\Policies\{<GUID>}`) regroupant une hiérarchie de fichiers qui contiennent les paramètres, scripts ou exécutables à appliquer.

L'analyse des GPO devra donc prendre en compte les deux parties : objets de l'annuaire et fichiers du Sysvol [4].

Enfin, une GPO sépare les paramètres destinés aux machines et aux utilisateurs, à la fois dans l'annuaire, où l'on retrouve deux sous-conteneurs `Machine` et `User` pour chaque objet de GPO, et dans le Sysvol, où l'on

retrouve deux dossiers du même nom pour chaque répertoire racine contenant les fichiers de GPO. Les attributs `GPC-Machine-Extension-Names` et `GPC-User-Extension-Names` permettent de savoir si une GPO s'adresse aux machines, aux utilisateurs, ou aux deux.

2.4 Contrôle d'accès

Active Directory repose sur le contrôle d'accès discrétionnaire des systèmes Windows, mais utilise des mécanismes particuliers afin de pouvoir l'étendre à volonté.

2.4.1 Contrôle d'accès des systèmes Windows

Le modèle de contrôle d'accès Windows permet d'établir des permissions d'accès pour tous les objets regroupés sous le terme générique *securable object*. Il peut s'agir par exemple d'éléments du système de fichiers ou de la base de registre, d'objets noyau, ou dans le cas d'AD, d'objets de l'annuaire. Les permissions positionnées profitent à des entités regroupées sous le terme *Security Principals* : utilisateurs, machines, groupes, ou entités spécifiques comme « Tout le monde » ou encore « Utilisateurs authentifiés ». Ils sont représentés par un identifiant de sécurité, nommé SID, pouvant respecter un format particulier :

- en environnement Active Directory, les SID commencent par l'identifiant du domaine, suivi d'une partie relative, le RID, propre à l'entité ;
- en environnement local, ils commencent par le SID de la machine locale, toujours suivi du RID ;
- certains SID sont dits *well-known* et restent les mêmes ou gardent le même RID d'un système à un autre [21].

Les permissions associées à un *securable object* sont stockées dans son **descripteur de sécurité**. Il contient notamment le propriétaire de l'objet et une DACL (*Discretionnary Access Control List*) composée d'ACE (*Access Control Entries*), où chaque ACE décrit une ou plusieurs permissions accordées ou refusées à un *security principal*.

Une ACE est principalement composée de trois éléments :

- un en-tête, contenant des informations de type et d'héritage des permissions ;
- un *trustee* (ou bénéficiaire), représenté par son SID ;
- un *access-mask*, où chaque bit désigne une permission (accordée ou refusée, selon le type d'ACE).

Si ces ACE sont utilisables de manière générique sur un grand nombre d'objets de types différents, elles sont néanmoins spécifiques : leur structure est générique, mais la sémantique des droits positionnés par l'ACE est spécifique à l'objet sur lequel elle s'applique. En effet, un *access-mask* est composé de différents types de droits (cf. figure 3) :

- des **droits génériques**, utilisés dans certains cas particuliers et associés à un ensemble d'autres droits (standards ou spécifiques) formant des opérations communes (`GENERIC_READ`, `GENERIC_WRITE`, `GENERIC_EXECUTE`, ou `GENERIC_ALL`) ;
- des **droits standards**, valables pour tous les types d'objets (5 sont actuellement définis) : `DELETE`, `READ_CONTROL`, `WRITE_DAC`, `WRITE_OWNER` et `SYNCHRONIZE` ;
- des **droits spécifiques**, valables pour le type d'objet courant uniquement (16 au maximum pour chaque type d'objet). Par exemple, le 6ème bit de poids faible du masque d'accès d'une ACE (valeur `0x00000020`) peut désigner le droit `FILE_EXECUTE` si l'ACE est positionnée sur un fichier, tout comme il peut représenter le droit complètement différent `DS_WRITE_PROP` si elle est positionnée sur un objet d'un annuaire AD.

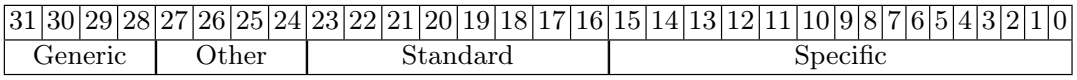


Figure 3. Format des masques d'accès utilisés par les ACE : droits génériques, standards et spécifiques. Les droits marqués comme « Other » sont `ACCESS_SYSTEM_SECURITY`, `MAXIMUM_ALLOWED` ou réservés pour usage futur.

Enfin, même si le mécanisme discretionnaire présenté ici est la composante principale du contrôle d'accès des systèmes Windows, ce dernier met en œuvre d'autres mécanismes, notamment :

- un contrôle d'accès **mandataire** basé sur des niveaux d'intégrité et des politiques d'accès aux objets [2] ;
- un contrôle d'accès **dynamique** basé sur des ACE conditionnelles testant la valeur de *Claims* (ou revendications) associées aux utilisateurs et de *Resource-properties* [15].

2. Ainsi, il est possible de définir des politiques d'accès refusant à un *security principal* les accès en lecture, écriture ou exécution à un *securable object* de niveau d'intégrité supérieur (politiques `NO_READ_UP`, `NO_WRITE_UP`, `NO_EXECUTE_UP`) [17].

2.4.2 Extension à Active Directory

Le mécanisme de contrôle d'accès Windows est personnalisable. Ainsi, les objets d'annuaire AD peuvent à ce jour se voir affecter 9 droits spécifiques [10] :

- `DS_CREATE_CHILD` ;
- `DS_DELETE_CHILD` ;
- `ACTRL_DS_LIST` ;
- `DS_SELF` ;
- `DS_READ_PROP` ;
- `DS_WRITE_PROP` ;
- `DS_DELETE_TREE` ;
- `DS_LIST_OBJECT` ;
- `DS_CONTROL_ACCESS`.

Ce modèle ne convient pas aux objets de l'annuaire sur deux points :

- sa **granularité** est insuffisante, car il est nécessaire de pouvoir placer des permissions d'accès en lecture et écriture, non pas sur l'objet complet, mais uniquement sur certaines de ses propriétés bien précises ;
- son **extensibilité** est limitée, puisqu'il n'est possible de définir que 16 droits spécifiques au maximum, limitation imposée par la taille des masques d'accès, alors qu'il est nécessaire de définir des droits pour un grand nombre de classes d'objets différentes, ainsi que de nouveaux droits pour des classes d'objets arbitraires inconnues de Microsoft, puisque le schéma AD peut être étendu par des applications tierces.

Ainsi, Active Directory étend le modèle de contrôle d'accès en utilisant les ACE de type `*_OBJECT_ACE`. Ces ACE étendent la structure précédemment détaillée en rajoutant notamment un champ `ObjectType` dont la sémantique dépend du cas d'utilisation (cf. figure 4). Les droits spécifiques AD utilisant ce type d'ACE sont : `DS_CREATE_CHILD`, `DS_DELETE_CHILD`, `DS_SELF`, `DS_READ_PROP`, `DS_WRITE_PROP` et `DS_CONTROL_ACCESS`.

« STANDARD » ACE :

Header			Access Mask	Trustee
Type	Flags	Size		

« OBJECT » ACE :

Header			Access Mask	Object Type	Inherited Object Type	Trustee
Type	Flags	Size				

Figure 4. Différences de structures entre ACE « classiques » et ACE « objets ».

Il est alors intéressant de détailler les droits spécifiques suivants :

- **les lectures/écritures de propriétés** (ou groupes de propriétés) : les droits spécifiques `DS_READ/WRITE_PROP` permettent de positionner des permissions d'accès en lecture/écriture sur une propriété particulière d'un objet. La propriété ciblée est spécifiée dans l'`ObjectType` par son GUID, présent dans l'attribut `schemaIDGUID` de l'entrée de la partition schéma de classe `attributeSchema` qui la représente. Ainsi, `bf9679e3-0de6-11d0-a285-00aa003049e2` représente la propriété `NT-Security-Descriptor`, décrite par l'objet `CN=NT-Security-Descriptor,CN=Schema,CN=Configuration,-DC=<domaine>`. De même, il existe des ensembles de propriétés prédéfinis, permettant de positionner ce type de permissions pour un groupe de propriétés rassemblées autour d'un même contexte (*Personal-Information*, *Email-Information*, *User-Account-Restrictions*, etc.) ;
- **les écritures validées** : le droit spécifique `DS_SELF` permet d'effectuer une « écriture de propriété validée ». Il s'agit d'une écriture de propriété particulière, pour laquelle des vérifications ont lieu sur la valeur écrite ou sur le *trustee*. L'opération à réaliser est spécifiée dans l'`ObjectType` par son GUID, présent dans l'attribut `schemaIDGUID` de l'entrée de la partition configuration de classe `controlAccessRight` qui la représente. Par exemple, `bf9679c0-0de6-11d0-a285-00aa003049e2` représente l'écriture validée `Self-Membership` décrite par l'objet `CN=Self-Membership,-CN=Extended-Rights,CN=Configuration,DC=<domaine>`. Elle permet l'édition de la propriété `Member` d'un objet de classe `Group`, mais uniquement avec le DN de l'utilisateur bénéficiaire du droit, autrement dit, elle permet à un utilisateur de s'ajouter ou se retirer d'un groupe ;
- **les droits étendus** : le droit spécifique `DS_CONTROL_ACCESS` permet d'effectuer toute autre action dont la sémantique n'est pas liée à des propriétés particulières des objets ciblés, ou pour laquelle un contrôle d'accès non supporté par le modèle standard est nécessaire. Une telle action, appelée « droit étendu », est ici aussi spécifiée dans l'`ObjectType` par son GUID, présent dans l'attribut `schemaIDGUID` de l'entrée de la partition configuration de classe `controlAccessRight` qui la représente. Par exemple, `45ec5156-db7e-47bb-b53f-dbeb2d03c40f` représente le droit étendu `Reanimate-Tombstones` décrit par l'objet `CN=Reanimate-Tombstones,CN=Extended-Rights,-`

`CN=Configuration,DC=<domaine>`. Il permet de restaurer des objets supprimés, mais tout de même conservés pendant une période, ce qui serait compliqué à mettre en place avec le contrôle d'accès standard.

Enfin, le mécanisme d'`ObjectType` peut aussi permettre de restreindre les permissions accordées à un type d'objet particulier en spécifiant le GUID d'une classe d'objets. Les permissions accordées ne le seront alors que si le bénéficiaire est une instance de cette classe [3]. Par exemple, l'ACE suivante est tirée d'un domaine de niveau fonctionnel `2008R2`, sur lequel Microsoft Exchange est installé. Elle fait partie du descripteur de sécurité de l'objet particulier `adminSDHolder`, qui sera appliqué à plusieurs types d'objets différents (cf. partie 2.5).

```
Ace Type:  0x5 - ACCESS_ALLOWED_OBJECT_ACE_TYPE
Ace Size:  56 bytes
Ace Flags: 0x2
    CONTAINER_INHERIT_ACE
Object Ace Mask:  0x000f01ff
    DELETE
    READ_CONTROL
    WRITE_DAC
    WRITE_OWNER
    ACTRL_DS_CREATE_CHILD
    ACTRL_DS_DELETE_CHILD
    ACTRL_DS_LIST
    ACTRL_DS_SELF
    ACTRL_DS_READ_PROP
    ACTRL_DS_WRITE_PROP
    ACTRL_DS_DELETE_TREE
    ACTRL_DS_LIST_OBJECT
    ACTRL_DS_CONTROL_ACCESS
Object Ace Flags: 0x1
    ACE_OBJECT_TYPE_PRESENT
Object Ace Type:  msExchDynamicDistributionList - 018849b0-a981-11d2
    -a9ff-00c04f8eedd8
Object Ace Sid:   CN=Exchange Trusted Subsystem,OU=Microsoft
    Exchange Security Groups,DC=domain [S
    -1-5-21-1234567890-1234567890-1234567890-1234]
```

L'`ObjectType` utilisé ici désigne la classe `msExchDynamicDistributionList`. Ainsi, les différents droits positionnés par cette ACE (*full control* pour le bénéficiaire *Exchange Trusted Subsystem*), ne le seront effectivement que pour les instances de cette classe. Les autres objets auxquels s'applique ce descripteur de sécurité, en particulier des objets très privilégiés du domaine comme notamment

3. Pour un objet particulier, l'`ObjectType` peut en réalité représenter n'importe quel élément faisant partie d'un arbre construit à partir de la classe d'objet, de groupes de propriétés et d'attributs.

le groupe d'administrateurs du domaine, ne seront pas accessibles par le bénéficiaire *via* cette ACE.

2.5 Gestion des comptes

La gestion des comptes du domaine regroupe de nombreux sujets : nomenclature des comptes, comptes de service, comptes d'administration, comptes obsolètes, etc. Pour notre analyse, il est nécessaire d'inspecter :

- la hiérarchie d'appartenance aux groupes de sécurité des utilisateurs, machines et sous-groupes, afin d'identifier les comptes appartenant à des groupes privilégiés, d'évaluer la taille de ces groupes privilégiés, etc. ;
- les attributs de sécurité des comptes utilisateurs, qui peuvent premièrement indiquer le « niveau d'hygiène » du domaine (comptes désactivés, comptes obsolètes, mots de passe n'expirant jamais, etc.), mais aussi avoir un impact plus fort sur sa sécurité, avec par exemple des attributs tels que :
  - `SID-History`, qui contient des SID présumés d'anciens domaines, conservés après une migration afin de préserver l'accès aux anciennes ressources importées. Ces SID seront ajoutés au jeton d'accès de l'utilisateur, et peuvent donc représenter une appartenance masquée à un groupe s'il s'agit de SID du domaine courant,
  - `Admin-Count`, positionné pour chaque utilisateur appartenant à un compte natif privilégié, protégé par le mécanisme d'`adminSDHolder` et de `SDProp`. Ce mécanisme sert à protéger ces comptes vis-à-vis d'eux-mêmes et à les empêcher de se retirer des permissions nécessaires à l'administration du domaine, en leur réappliquant périodiquement le descripteur de sécurité de l'`adminSDHolder`. Même quand un compte n'appartient plus à un groupe protégé, l'attribut `Admin-Count` demeure, et il a notamment pour effet de bloquer l'héritage des permissions, ce qui peut empêcher l'application de restrictions de droits.

La gestion des comptes du domaine est donc une tâche complexe, alourdie par la difficulté d'identifier les groupes et comptes **non-natifs** privilégiés. En effet, Microsoft définit une liste de comptes et groupes *builtin* privilégiés [12], mais d'autres que ceux-ci peuvent faire l'objet de délégation de droits sur des conteneurs et objets du domaine. Pour les révéler, une analyse des permissions est donc obligatoire.

3 Relations de contrôle direct

Les relations de contrôle constituent le cœur de notre méthode d'analyse. Elles seront ensuite agrégées pour former des chemins de contrôle. Ainsi, ces chemins seront d'autant plus complexes qu'il existe un nombre important de relations différentes à relever, et qu'elles touchent un nombre varié d'objets du domaine. Cette partie détaille un ensemble de relations exploitables dans le cadre de la démarche d'analyse présentée.

3.1 Définition

Les **relations de contrôle direct** traduisent la maîtrise totale d'un objet de l'AD sur un autre : du fait de la présence d'une telle relation, on considère qu'il n'existe plus de barrière de sécurité de l'objet maître vers l'objet esclave. Nous appellerons l'agrégation d'une suite de ces relations entre objets un **chemin de contrôle**. Par transitivité, le contrôle du premier objet du chemin permet indirectement le contrôle de tous les objets du chemin.

Certaines relations sont immédiates, comme la permission de changer le mot de passe d'un utilisateur sans en connaître l'actuel, alors que d'autres sont différées et demandent une action particulière de la part de l'objet visé. Par exemple, le droit de réécriture d'un fichier de stratégie de groupe, qui nécessite, afin de contrôler l'utilisateur ou la machine cible, que la stratégie en question soit réappliquée (manuellement ou après un redémarrage).

On peut donc considérer les relations et chemins de contrôle comme des transitions dont la mise en œuvre est plus ou moins complexe d'un contexte de sécurité vers un autre. Une telle transition peut représenter une vulnérabilité dès lors qu'elle permet d'atteindre un contexte de sécurité plus privilégié, de manière directe ou par rebonds successifs.

3.2 Relations considérées

Sont décrits dans cet article plusieurs types de relations de contrôle, sur un ensemble d'objets manipulés par Active Directory : objets de l'annuaire, stratégies de groupes, groupes et comptes locaux, journaux des machines locales, etc. Cette partie présente ces relations, leur provenance (descripteur de sécurité, attribut de classe particulier, etc.), ainsi que les moyens de les relever. Le formalisme utilisé pour les modéliser et les représenter sera présenté en partie 4.

3.2.1 Relations « génériques » sur tout type de *securable object*

Un très grand nombre de relations de contrôle sont liées aux descripteurs de sécurité. Étant donné qu'une partie du contrôle d'accès discrétionnaire des systèmes Windows est générique, les relations de contrôle suivantes sont génériques et valables sur plusieurs types d'objets. Les éléments suivants confèrent une relation de contrôle sur l'objet auquel appartient le descripteur de sécurité :

- **le propriétaire de l'objet**, puisqu'il possède implicitement le droit `WRITE_DAC` décrit ci-dessous ;
- **le droit standard** `WRITE_DAC`, puisqu'il permet de modifier le contrôle d'accès discrétionnaire lié à l'objet. Ainsi, il est possible de rajouter une ACE dans la DACL de l'objet et de s'accorder par exemple son contrôle total ;
- **le droit standard** `WRITE_OWNER`, puisqu'il permet de modifier le propriétaire de l'objet et donc de se ramener au premier cas présenté ;
- **le droit générique** `GENERIC_WRITE` **ou le droit générique le contenant** `GENERIC_ALL`**.** Bien que pouvant être mappés à un ensemble de droits différents selon les types d'objets, ces droits peuvent être considérés comme établissant des relations de contrôle étant donné le large éventail d'opérations en écriture qu'ils permettent. Par exemple pour les objets de l'annuaire, le premier autorise entre autres toutes les écritures de propriétés et toutes les écritures validées, et le second rajoute notamment `WRITE_DAC` et `WRITE_OWNER`.

À cela on peut ajouter **l'absence de DACL sur un objet** [4], puisque cela signifie que tous les accès, quels qu'ils soient, sont autorisés pour tous les demandeurs. Étant donné que cette relation ne désigne pas un bénéficiaire particulier, il est possible d'utiliser l'entité « Tout le monde » comme source de la relation, réprésentée par le SID `S-1-1-0`.

3.2.2 Relations sur tous les objets de l'annuaire

On pourra également établir une relation de contrôle de manière générique sur un objet de l'annuaire, en présence d'ACE conférant un niveau important de droits, à savoir :

- **les ACE autorisant l'écriture de toutes les propriétés,** c'est-à-dire le droit `DS_WRITE_PROP`, sans `ObjectType`.
- **les ACE autorisant tous les droits étendus,** c'est-à-dire le droit `DS_CONTROL_ACCESS`, sans `ObjectType`.

4. À ne pas confondre avec une DACL vide, qui a pour effet de refuser tous les accès, sauf les implicites pour le propriétaire de l'objet ou les possesseurs du privilège `SeTakeOwnershipPrivilege` [9].

Étant donné le faible nombre d'écritures validées différentes, il n'est pas possible d'établir le contrôle de manière systématique en présence d'ACE autorisant toutes les écritures validées (droit `DS_SELF` sans `ObjectType`). Elles seront néanmoins intéressantes pour les objets de type `Group`.

3.2.3 Relations particulières de hiérarchie

Avant de décrire les différentes relations propres à chaque type d'objet, il est nécessaire d'en décrire deux types particuliers n'étant pas exactement des relations « de contrôle », mais qu'il est nécessaire de considérer comme telles. Ces relations sont des **relations de hiérarchie**, servant à lier entre eux les différents objets, afin que le contrôle des objets soit propagé à des objets sous-jacents. On considère notamment :

- **l'appartenance à un groupe de sécurité**, qui constitue une relation du membre vers le groupe. En effet, un groupe est un *security principal* pouvant bénéficier de permissions accordées par des ACE. Ses membres bénéficieront également de ces permissions, car le SID du groupe sera inclus dans leur « jeton d'accès » (*Access Token*). Ainsi, plutôt que de dupliquer les relations de contrôle qu'un groupe possède à chacun de ses membres, il est préférable de placer une relation de chaque membre vers le groupe, comme illustré sur la figure 5.

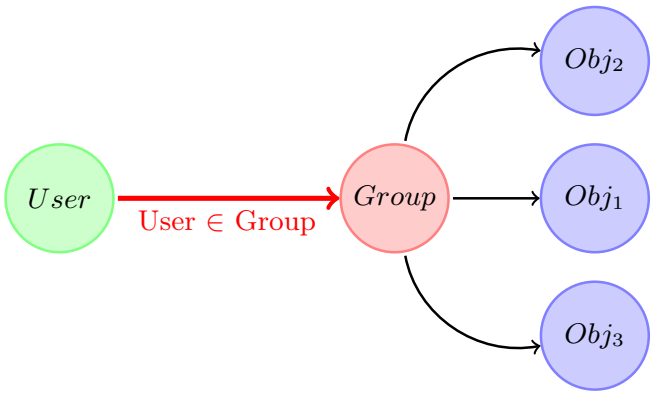


Figure 5. Contrôle d'un utilisateur sur plusieurs objets *via* son appartenance à un groupe.

- **l'appartenance à un conteneur**, comme une unité organisationnelle, qui constitue une relation du conteneur vers ses enfants, due au mécanisme d'héritage des permissions. Par défaut, les ACE héritables placées sur un conteneur sont dupliquées dans les descripteurs

de sécurité des objets fils [5]. Ainsi, quand la hiérarchie de l'annuaire devient complexe, l'ajout d'une seule ACE peut faire augmenter fortement le nombre d'ACE total dû à cette duplication. De plus, il n'est pas utile d'analyser de multiples fois une même ACE. Ainsi, tracer une relation de chaque conteneur vers ses fils (comme ilustré sur la figure 6) permet de réduire radicalement la quantité d'ACE à inspecter [6], en supprimant les ACE héritées pour ne garder que celles étant propres à un objet. Les relations de contrôles dues aux ACE héritées sont donc matérialisées par ces liens de transitivité partant des conteneurs sur lesquels elles s'appliquent.

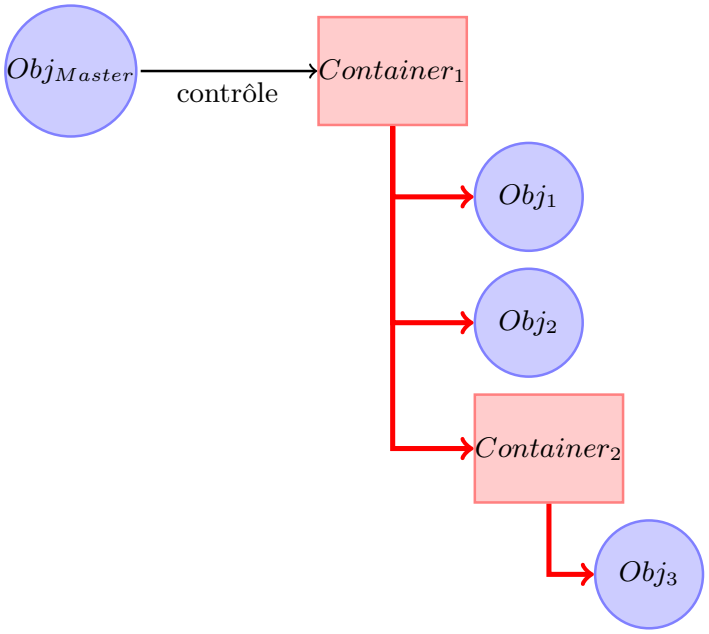


Figure 6. Propagation (en gras/rouge) du contrôle d'un objet aux enfants de conteneurs, *via* l'héritage des droits.

3.2.4 Relations sur les conteneurs LDAP

Différents conteneurs LDAP [7], comme `Organizational-Unit`, `Domain` et

5. Par défaut, car les mécanismes d'héritage dépendent de nombreux éléments et sont de fait complexes (par exemple, un descripteur de sécurité peut être paramétré pour bloquer l'héritage).

6. Les ACE héritées peuvent représenter parfois plus de 95% des ACE présentes sur les objets de l'annuaire.

7. Le terme « conteneur LDAP » désigne ici de manière générale tout objet pouvant posséder des objets fils dans l'annuaire, à ne pas confondre avec les objets de classe `Container`.

`Domain-DNS`, ou encore `Group-Policy-Container` sont également la cible de relations de contrôle. On considère notamment :

- **l'écriture de propriété** `GP-Link`, c'est-à-dire le droit `DS_WRITE_PROP` et l'`ObjectType` `f30e3bbe-9ff0-11d1-b603-0000f80367c1`, qui permet de lier une GPO existante à un conteneur, et ainsi d'exécuter la GPO sur les utilisateurs et machines membres de ce conteneur en correspondance avec les paramètres de GPO définis ;
- **le droit étendu** `DS-Replication-Get-Changes-All` sur les objets de classe `Domain-DNS`, c'est-à-dire le droit `DS_CONTROL_ACCESS` et l'`ObjectType` `1131f6ad-9c07-11d1-f79f-00c04fc2dcd2`, peut être vu comme un moyen de contrôle sur tout le domaine puisqu'il permet d'obtenir les attributs secrets [13] comprenant notamment les empreintes des mots de passe des utilisateurs [8] ;
- **les données de la propriété** `GP-Link` **des conteneurs**, qui permettent d'établir une relation de contrôle entre des GPO liées et un conteneur auquel elles sont liées. Cette propriété suit le format suivant :

```
[<GPO-DN>;<GPLink-Options>]+
```

  Les champs `GPO-DN` représentent les GPO liées, et `GPLink-Options` un masque d'options permettant de spécifier si la GPO est *Enforced* [9] (valeur `0x2`) ou *Ignored* (valeur `0x1`). On ajoutera donc une relation de contrôle pour toute GPO liée non ignorée.

3.2.5 Relations sur les objets `Group`

L'appartenance à un groupe de sécurité peut être considérée comme une relation de contrôle (comme défini dans la partie 3.2.3). Ainsi, les permissions permettant de modifier les appartenances à un groupe, ou les propriétés représentant des appartenances cachées sont des relations de contrôle. On considère notamment :

- **l'écriture de propriété** `Member`, c'est-à-dire le droit `DS_WRITE_PROP` et l'`ObjectType` `bf9679c0-0de6-11d0-a285-00aa003049e2`. Cette propriété permet de modifier les membres d'un groupe, et donc de s'y ajouter ;

8. À ne pas confondre avec les autres droits étendus `DS-Replication-*` ne permettant pas de récupérer les attributs secrets.

9. Quand une GPO est *Enforced*, ses paramètres ne peuvent pas être redéfinis par une autre GPO liée à un sous-conteneur (qui serait normalement prioritaire).

- **l'écriture du groupe de propriétés** `Membership`, c'est-à-dire le droit `DS_WRITE_PROP` et l'`ObjectType` `bc0ac240-79a9-11d0-9020-00c04fc2d4cf`, puisque ce groupe contient notamment la propriété `Member` ;
- **l'écriture validée** `Self-Membership`, c'est-à-dire le droit `DS_SELF` et l'`ObjectType` `bf9679c0-0de6-11d0-a285-00aa003049e2`, puisqu'elle permet elle aussi de s'ajouter comme membre du groupe visé ;
- **les ACE autorisant toutes les écritures validées,** c'est-à-dire le droit `DS_SELF`, sans `ObjectType`, puisqu'elles permettent entre autres l'écriture validée `Self-Membership` nous ramenant au cas précédent ;
- **les données de la propriété** `SID-History`, puisqu'elle contient des SID qui seront rajoutés au jeton d'accès des utilisateurs, et peuvent donc représenter une appartenance cachée à un groupe du domaine si le SID du groupe est présent dans le `SID-History` d'un utilisateur ou d'un groupe ;
- **les données de la propriété** `Primary-Group-ID`, puisqu'elles représentent le RID du groupe principal d'un utilisateur, qui n'est pas listé dans les propriétés `Member` et `MemberOf` [11] [10].

3.2.6 Relations sur les objets `User`

Les permissions et propriétés suivantes constituent des relations de contrôle sur les objets de classe `User` (et donc possiblement aussi sur les objets de classe `Computer`, qui hérite de la classe `User`). On considère notamment :

- **le droit étendu** `User-Force-Change-Password`, c'est-à-dire le droit `DS_CONTROL_ACCESS` et l'`ObjectType` `ab721a53-1e2f-11d0-9819-00aa0040529b`, puisqu'il permet de modifier le mot de passe d'un objet utilisateur sans en connaître l'actuel [11] ;
- **l'écriture de propriété** `Script-Path`, c'est-à-dire le droit `DS_WRITE_PROP` et l'`ObjectType` `bf9679a8-0de6-11d0-a285-00aa003049e2`. Cette propriété contient un chemin vers un script exécuté à l'ouverture de la session

10. Les comptes dont le `Primary-Group-ID` ne contient pas la valeur par défaut devront faire l'objet d'une attention particulière (« 513 » représentant le groupe *Domain Users* pour les utilisateurs et « 515 » représentant le groupe *Domain Computers* pour les comptes de machines).

11. À ne pas confondre avec le droit étendu `User-Change-Password`, accordé à tout le domaine, qui nécessite de connaître le mot de passe courant pour en changer.

d'un utilisateur. Elle référence habituellement un script situé dans le partage `NETLOGON` [12], mais peut aussi contenir un chemin UNC (*Uniform Naming Convention*) vers un partage contrôlé par un attaquant.

Plusieurs autres accès en lecture à des propriétés pourraient représenter une relation de contrôle sur les objets utilisateurs. Par exemple la lecture des attributs `dBCSPwd` et `unicodePwd` contenant les empreintes LM et NT du mot de passe de l'utilisateur. Cependant, les accès à ces attributs sont inconditionnellement refusés, peu importe le contrôle d'accès mis en place [8].

Enfin, d'autres relations sur les objets `User` et `Computer` liées aux postes locaux sont présentées ci-après en partie 3.2.8.

3.2.7 Relations liées aux stratégies de groupe

Les stratégies de groupe sont également sources de relations de contrôle. En effet nous avons déjà pu voir que nous considérions la permission d'écriture de la propriété `GP-Link` comme une relation de contrôle sur un conteneur, car permettant de lui lier une GPO. Les GPO étant séparées en deux parties distinctes, des relations de contrôle sur la GPO peuvent provenir de chacune des deux parties : objets de GPO dans l'annuaire et fichiers de GPO dans le Sysvol.

Concernant les objets de GPO dans l'annuaire, on retrouve les relations génériques et celles s'appliquant à tout type d'objet de l'annuaire ou aux objets de type conteneurs (car les objets de GPO sont de classe `Group-Policy-Container` qui hérite de la classe `Container`). On peut également ajouter :

- **l'écriture de propriété** `GPC-File-Sys-Path`, c'est-à-dire le droit `DS_WRITE_PROP` et l'`ObjectType` `f30e3bc1-9ff0-11d1-b603-0000f80367c1`. Cette propriété contient le chemin UNC du répertoire racine contenant les fichiers de GPO, normalement sur le Sysvol. La modifier permet de fixer ce chemin vers un répertoire contrôlé, pouvant se situer en dehors du Sysvol, contenant des fichiers de GPO malveillants.

Concernant le Sysvol, les fichiers de GPO peuvent avoir une arborescence complexe. Comme le montre la figure 7, ils sont organisés par type pour chacun des dossiers utilisateurs et machines : paramètres de sécurité, clés de registre, conteneurs d'applications (AAS - *Application Advertise Script*), scripts, modèles d'administration, redirections de dossiers, etc.

12. `\\<domaine>\SYSVOL\<domaine>\Scripts`.

```
{GUID}\
    User\
        Applications\
        Documents and Settings\
        Microsoft\
            IEAK\
            Remote Install\
        Scripts\
            Logon\
            Logoff\
    Machine\
        Adm\
        Applications\
        Microsoft\
            Windows NT\
                Secedit\
        Scripts\
            Startup\
```

Figure 7. Arborescence type d'un répertoire de GPO.

Il est donc nécessaire de définir quels sont les fichiers ou dossiers qui pourront, s'ils sont modifiables, conférer le contrôle de la GPO, et donc des utilisateurs ou machines auxquels elle s'applique [13]. Cela ne peut pas être réalisé de manière exhaustive, puisque des éléments tiers peuvent définir des moteurs d'application de paramètres de GPO (CSE - *Client Side Extensions*) pouvant s'appuyer sur n'importe quels éléments. On peut néanmoins commencer par les éléments génériques suivants :

- le répertoire racine de la GPO (nommé d'après son GUID), puisqu'il contient toute la GPO ;
- les deux sous-répertoires `User` et `Machine`, puisqu'ils contiennent tous les paramètres destinés respectivement aux utilisateurs et machines auxquels s'applique la GPO ;
- le fichier `Registry.pol`, situé dans les sous-dossiers `User` et `Machine`, car il permet de modifier toute la base de registre ;
- le sous-répertoire `Applications` et les fichiers `*.aas` qu'il contient, situé dans les sous-dossiers `User` et `Machine`, puisqu'il s'agit du répertoire contenant les applications à installer (au format AAS, embarquant des fichiers exécutables) ;
- le sous-répertoire `Scripts`, situé dans les sous-dossiers `User` et `Machine`, ainsi que ses sous-éléments : les répertoires `Logon` et `Logoff`

13. Le contrôle total de la GPO n'est pas nécessaire pour contrôler les utilisateurs et machines auxquels elle s'applique. Certains fichiers seuls sont suffisants.

pour les utilisateurs, `Startup` et `Shutdown` pour les machines, le fichier `Scripts.ini` qu'il contient, contenant les scripts à lancer, ainsi que tous les scripts présents référencés par ce dernier ;
- les sous-répertoires composant le chemin `Microsoft\WindowsNT-\Secedit\` situés dans le sous-dossier `Machine`, ainsi que le fichier `GptTmpl.inf` qu'il contient, contenant un grand nombre de paramètres de sécurité, par exemple les bénéficiaires des privilèges locaux sur la machine.

Pour ces fichiers et dossiers définis comme critiques, on pourra considérer qu'ils permettent de contrôler la GPO dans les conditions suivantes (les droits spécifiques sont ici ceux ayant un sens pour des fichiers et dossiers) :
- la présence de relations génériques présentées précédemment (dues à des droits ne dépendant pas du type d'objet, tels que `WRITE_DAC`, `WRITE_OWNER`, etc.) ;
- **les droits spécifiques `FILE_WRITE_DATA` et `FILE_APPEND_DATA`** (nommés `FILE_ADD_FILE` et `FILE_ADD_SUBDIRECTORY` pour les répertoires) puisqu'ils permettent de modifier les fichiers ou dossiers désignés, et donc de détourner leur comportement : selon les cas, ajouter des scripts à exécuter, redéfinir des paramètres de sécurité ou des clés de registres, etc.

Enfin, les relations de contrôle dues aux GPO ne sont pas immédiates. Elles nécessitent d'être réappliquées (manuellement ou automatiquement de manière périodique ou après un redémarrage) pour prendre le contrôle d'un objet auquel elles s'appliquent (utilisateur ou machine).

3.2.8 Relations liées aux postes locaux

Les relations définies jusqu'ici ne permettent pas de représenter un type de lien primordial : celui entre un utilisateur et les machines qu'il utilise. En effet, de nombreuses vulnérabilités dans la gestion du domaine découlent de l'utilisation de comptes privilégiés du domaine sur des postes de travail. Il est donc nécessaire d'ajouter les relations suivantes traduisant cette utilisation :
- **la connexion d'un utilisateur sur une machine** établit une relation de la machine sur l'utilisateur. En effet, la machine « maîtrise » l'utilisateur, dans le sens où un attaquant ayant obtenu les privilèges d'administration sur la machine peut compromettre l'utilisateur (en récupérant le cache des mots de passe, en utilisant un *keylogger*, etc.). Les événements de connexion des utilisateurs sur une machine peuvent être récupérés *via* le *provider* d'événements `Microsoft--Windows-TerminalServices-LocalSessionManager` : l'événement

portant l'`eventID` 21, `EVENT_SESSION_LOGON`, indique l'ouverture d'une session locale.

Ainsi, les *security principals* du domaine contrôlent une machine si :

- **ils sont membres des groupes locaux d'administration** de la machine. On pourra notamment considérer les groupes locaux `Administrators` et `Backup Operators` ;
- **ils possèdent un privilège local d'administration** sur la machine. On pourra notamment considérer les privilèges `SeTcbPrivilege`, `SeBackupPrivilege`, `SeRestorePrivilege`, `SeDebugPrivilege`, `SeCreateTokenPrivilege`, `SeLoadDriverPrivilege` ou `SeTakeOwnershipPrivilege`. Cependant, cette relation ne peut être établie sans condition que sur les systèmes pré-NT6. À partir de Windows *Vista* ces privilèges ne peuvent être accordés qu'aux utilisateurs possédant un niveau d'intégrité au minimum `High` dans leur jeton d'accès (c'est à dire aux comptes locaux de services et membres des groupes locaux `Administrators`, `Backup Operators`, `Network Configuration Operators` et `Cryptographic Operators` [23]).

3.2.9 Relations conditionnelles

Il est également possible de définir des relations de contrôle particulières, dont l'existence dépend de la configuration du domaine. Par exemple :

- si un domaine utilise l'authentification par *SmartCard*, on peut ajouter l'écriture des propriétés `User-Principal-Name`, `Service-Principal-Name` [1] et du groupe de propriétés `Public-Information` les contenant ;
- des paramètres de configuration tels que `DS-Heuristics` peuvent changer des règles d'utilisation ou d'accès à certains attributs. Certains peuvent peut-être définir de nouvelles relations ;
- des systèmes ou applications tiers peuvent utiliser des propriétés particulières de l'annuaire, n'ayant normalement pas d'importance. Par exemple la partie *primary group* des descripteurs de sécurité, qui peut être utilisée si l'annuaire est utilisé comme service LDAP pour des systèmes UNIX, l'attribut `User-Password`, qui peut être utilisé par des applications tierces pour enregistrer un mot de passe, ou encore l'attribut `Managed-By` qui peut désigner selon les cas un gestionnaire de groupe, un administrateur de RODC, un gestionnaire de liste de diffusion Exchange, et qui peut être source de données pour différents scripts d'administration ou applications [14] ;

14. `http://blogs.technet.com/b/askds/archive/2011/06/24/friday-mail-sack-wahoo-edition.aspx#managedby`.

– si des ACE conditionnelles [15] sont utilisées sur des ressources du domaine, des relations de contrôle peuvent être identifiées dans certains cas. Notamment si l'utilisateur possède le droit en écriture sur des propriétés désignées comme source de ses *claims*, inscrites dans les attributs `ms-DS-Claim-Source` des objets `ms-DS-Claim-Type` représentant ces *claims*, ou encore s'il possède le droit en écriture sur ce dernier attribut, ce qui lui permettrait de changer cette source afin qu'une *claim* soit liée à un attribut qu'il contrôle.

Nous venons de décrire l'ensemble minimal des relations de contrôle évidentes à considérer lors d'un audit. La dernière sous-partie, traitant des relations conditionnelles, démontre un point important : des relations peuvent dépendre de la configuration, de l'usage du domaine, et ne sont donc pas forcément décrites ici. De plus, toutes les propriétés, groupes de propriétés, droits étendus, écritures validées et classes d'objets n'ont pas pu être présentées de manière exhaustive et de nouveaux éléments seront assurément rajoutés par les prochaines mises à jour et extensions d'Active Directory. Le suivi et l'enrichissement de la liste d'objets et de relations considérés permettra d'étendre les analyses *via* la prise en compte de chemins de contrôle de plus en plus variés et complexes.

3.3 Méthodes de relevé

Nous avons essayé de définir des méthodes de relevé de ces relations de contrôle pouvant être utilisables quand c'est envisageable dans différents contextes. Par exemple pour l'annuaire, « à froid » (depuis une copie de la base `ntds.dit`) ou depuis le système « à chaud », avec un compte privilégié ou non. L'utilisation d'un compte non-privilégié permet, entre autres, de produire des outils respectant le principe du moindre privilège ne perturbant pas le fonctionnement du domaine.

3.3.1 Relevé des relations sur les objets de l'annuaire

Pour les objets de l'annuaire, la méthode choisie est le **requêtage LDAP**. En effet, toutes les relations de contrôle identifiées sur les objets de l'annuaire sont accessibles *via* des attributs des différents objets, dont le principal est le descripteur de sécurité, disponible *via* l'attribut `NT-Security-Descriptor`. Cette méthode a pour avantage :

15. ACE représentées dans l'annuaire notamment par des objets de type `ms-DS-Claim-Type`. Elles sont supportées par Active Directory à partir du niveau fonctionnel Windows Server 2012 [15].

– de pouvoir fonctionner à chaud comme à froid, puisqu'il est possible d'exposer un annuaire LDAP à partir d'un fichier `ntds.dit` à l'aide de l'outil `dsamain` [14] :

```
dsamain /dbpath <ditfilepath> /ldapPort <number> /
    allowNonAdminAccess
```

– de pouvoir fonctionner avec un compte non-privilégié, simple membre du domaine. En effet, l'attribut `Default-Security-Descriptor` de la classe `Top` [20] (dont toutes les autres classes héritent) contient l'ACE suivante au format SDDL [16] :

```
(A;;RPLCLORC;;;AU)
```

Que l'on peut traduire en :

```
Ace Type:  0x0 - ACCESS_ALLOWED_ACE_TYPE
Ace Mask:  0x00020094
    READ_CONTROL
    ACTRL_DS_LIST
    ACTRL_DS_READ_PROP
    ACTRL_DS_LIST_OBJECT
Ace Sid:   AUTORITE NT\Authenticated Users [S-1-5-11]
```

Cette ACE que l'on retrouve à la racine de chaque partition et qui sera donc héritée par la quasi-totalité des objets, spécifie que n'importe quel utilisateur authentifié peut lire toutes les propriétés et les informations liées au contrôle d'accès de chaque objet, sauf interdiction explicite.

Cela nous permet donc de relever les données nécessaires à l'établissement des relations de contrôle *via* de simples requêtes LDAP. L'attribut `NT-Security-Descriptor` nécessite cependant d'utiliser un « contrôle LDAP » [17] particulier. En effet, par défaut la demande de cet attribut renvoie le descripteur de sécurité complet, comprenant 4 éléments : *owner*, *primary group*, DACL et SACL. Or, l'accès à la SACL d'un objet n'est accordé que si le droit `ACCESS_SYSTEM_SECURITY` est accordé, et ce dernier nécessite de posséder le privilège `SeSecurityPrivilege`, que seuls les administrateurs du domaine possèdent par défaut. Il est donc nécessaire d'utiliser le contrôle `LDAP_SERVER_SD_FLAGS_OID` [16], permettant d'exclure la SACL des informations demandées.

16. SDDL - *Security Descriptor Definition Language* [19].

17. Les « contrôles LDAP » sont un mécanisme d'extension du protocole LDAP par les différents clients et serveurs. Ils sont spécifiés lors d'une requête par un OID (*Object Identifier*) [7].

3.3.2 Relevé des relations sur les fichiers de GPO

La méthode de relevé des fichiers de GPO consiste simplement à copier ces fichiers en conservant les descripteurs de sécurité. Cela peut être réalisé à l'aide de l'utilitaire `Robocopy` [18] :

```
robocopy \\<domaine>\SYSVOL\<domaine>\Policies <destination> /E /B /
    COPY:DATSO
```

Ici aussi, cette méthode permet le traitement à chaud comme à froid, et est accessible à un utilisateur non privilégié, puisqu'on retrouve sur chaque dossier de GPO une ACE autorisant l'accès en lecture aux utilisateurs authentifiés, qui sera ensuite héritée par ses sous-éléments. Cependant, certains fichiers de GPO peuvent faire l'objet d'une interdiction explicite et ne pas être accessibles aux simples utilisateurs. Dans ce cas un compte d'administration est requis pour la copie, et une fois copiés les fichiers pourront être analysés avec l'utilisation du privilège `SeBackupPrivilege` permettant de s'affranchir du contrôle d'accès.

3.3.3 Relevé des relations propres aux postes locaux

Afin de relever les relations liées aux postes locaux, il est nécessaire de tous les interroger individuellement. Ainsi, il est nécessaire d'accéder :

- aux journaux d'événements afin d'extraire les événements `EVENT_SESSION_LOGON` du *provider* `Microsoft-Windows--TerminalServices-LocalSessionManager` indiquant l'ouverture d'une session locale, en utilisant par exemple les API `OpenEventLog` et `ReadEventLog` ;
- aux groupes locaux afin d'en extraire les membres, en utilisant par exemple l'API `NetUserGetLocalGroups` ;
- aux privilèges locaux afin d'en extraire les détenteurs, ce qui n'est malheureusement pas faisable *via* l'API Win32, il sera nécessaire d'analyser les données binaires stockées dans la base de registre dans les sous clés `ActSysAc` et `Privilgs` de la clé `\HKEY_LOCAL_MACHINE-\SECURITY\Policy\Accounts\<SID>` [2].

4 Chemins et visualisation

Le volume de données récoltées est souvent important : un relevé sur un domaine réel peut produire plusieurs dizaines de millions de relations. Il est nécessaire de les stocker dans un format adéquat pour pouvoir les exploiter. Dans cette partie, la modélisation utilisée pour analyser les relations de contrôle sera présentée. Ensuite, nous détaillerons une

implémentation adaptée utilisant une base de données orientée graphe. Enfin, nous présenterons une méthode de visualisation graphique des résultats de requêtes dans cette base de données.

4.1 Modélisation

L'ensemble des objets est représenté par des nœuds dans un graphe orienté. Une relation de contrôle d'un objet maître vers un objet esclave est représentée par un arc allant du maître vers l'esclave. Ainsi, dire qu'un objet A contrôle un objet B à travers une relation r est équivalent au graphe 8.

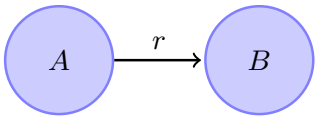


Figure 8. Relation de contrôle simple.

La relation de contrôle est transitive. Ainsi, si un objet A contrôle un objet B, qui lui-même contrôle un objet C, on peut en conclure que l'objet A contrôle l'objet C. Il est aussi possible qu'un objet A contrôle un objet C *via* plusieurs autres objets, B_1 et B_2, comme c'est le cas sur la figure 9. Ainsi, on peut dire qu'un objet A contrôle un objet B si et seulement si il existe au moins un chemin allant de A à B.

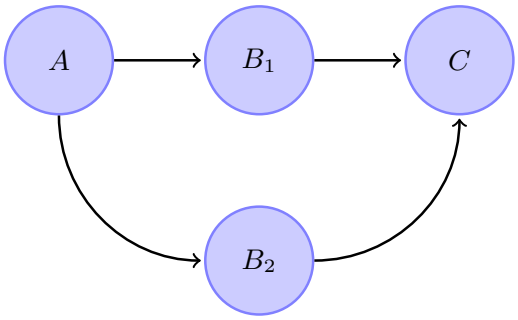


Figure 9. Contrôle *via* plusieurs objets.

On peut parcourir le graphe dans deux sens, depuis l'objet maître vers l'esclave, ou bien le contraire. Ainsi, il est possible de définir l'ensemble des objets contrôlant un objet B comme l'ensemble des objets A_i pour lesquels il existe un chemin de contrôle entre A_i et B. On peut aussi

définir l'ensemble des objets contrôlés par un objet A comme l'ensemble des nœuds B_i pour lesquels il existe un chemin de contrôle allant de A à B_i. Le cas le plus simple est celui d'un chemin de longueur 1 : les prédécesseurs d'un objet A le contrôlent directement, et l'objet A contrôle directement l'ensemble de ses successeurs. Ces points sont illustrés sur les figures 10 et 11.

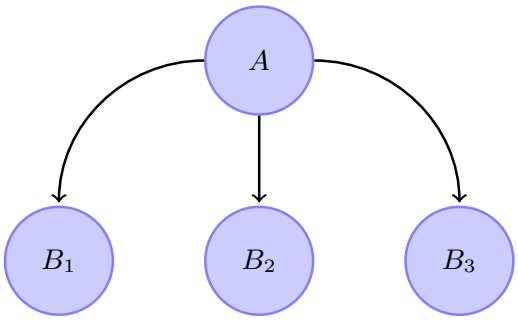


Figure 10. Successeurs d'un objet.

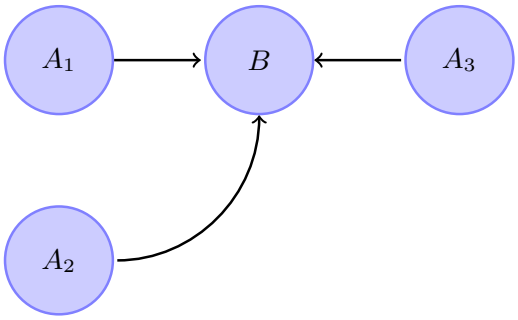


Figure 11. Prédécesseurs d'un objet.

Il peut bien entendu exister plusieurs arcs de même sens entre deux nœuds du graphe. Par exemple, un objet A ayant plusieurs ACE de contrôle sur un objet B est représenté sur la figure 12.

Chacun des différents arcs r_i représente une relation de contrôle différente. Dans la représentation choisie (décrite dans le paragraphe suivant), chaque arc possède un type décrivant la nature de cette relation.

En utilisant cette modélisation, la résolution de chemins de contrôle revient mathématiquement à résoudre des problèmes de connectivité dans un graphe orienté. On peut par exemple rechercher :

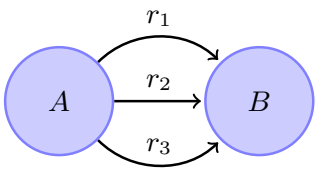


Figure 12. Relations multiples entre deux objets.

- l'ensemble des ressources de l'Active Directory que peut contrôler un compte utilisateur particulier ;
- l'ensemble des comptes utilisateur pouvant obtenir les privilèges d'administration du domaine ;
- les différents chemins de contrôle permettant à un compte particulier d'obtenir les privilèges d'administration de domaine ;
- des éventuelles portes dérobées, car un compte compromis aurait pu se laisser la possibilité d'obtenir de hauts privilèges *via* un chemin de contrôle non trivial et difficile à découvrir sans outil d'analyse.

4.2 Base de données orientée graphe

4.2.1 Présentation

La modélisation choisie pour représenter les relations de contrôle est celle d'un graphe ; l'utilisation d'un moteur de base de données orientée graphe est donc la solution naturelle pour stocker et exploiter les données. Il existe de nombreux moteurs différents. On peut citer par exemple FlockDB [5] (utilisé par Twitter), OrientDB [27] ou Neo4j [26]. C'est cette dernière solution qui a été choisie pour stocker et exploiter les relations de contrôle en environnement Active Directory. Neo4j offre des fonctionnalités qui permettent de mettre en œuvre la modélisation décrite dans le paragraphe précédent :

- un graphe (l'objet exploité par le moteur de base de données) est composé de nœuds, de relations et de propriétés ;
- un nœud possède un nombre arbitraires de propriétés, permettant de caractériser l'objet représenté (nom, type) ;
- une relation permet de modéliser un lien entre deux nœuds, et possède une direction et un nombre arbitraire de propriétés (nom, caractéristique de la relation).

Neo4j possède des performances satisfaisantes et permettant de traiter la grande quantité de données extraites de l'environnement Active Directory. De plus, il existe de nombreux moyens de s'interfacer avec le moteur :

- une console où on peut entrer des requêtes directement ;

– une interface de programmation officielle REST ;
– plusieurs librairies non officielles permettant d'interagir avec la base en utilisant des programmes Python, Ruby, PHP.

4.2.2 Import en base

L'import des données en base doit être rapide même pour des volumes importants de données. Neo4j offre deux méthodes d'import : à froid et à chaud. L'import à froid consiste à créer le fichier de base de données à part, sans que le service de base de données ne soit lancé. Une fois le fichier créé, il est installé dans l'arborescence Neo4j, puis on exécute le serveur. Cet import est très rapide et n'utilise pas le moteur transactionnel de Neo4j afin d'optimiser la vitesse d'exécution. Il est particulièrement adapté à un premier import de données dans une base vide. L'import à chaud est celui utilisé par les moteurs de base de données classiques : il est beaucoup plus lent, mais permet de modifier les données d'une base déjà créée.

Les auteurs ont développé un programme d'import à froid (dit *batch inserter*) Java afin de profiter au maximum des performances de Neo4j. L'API est décrite sur le site de Neo4j [24], et le développement d'un tel outil ne présente aucune difficulté. L'import en base de l'ensemble des relations extraites d'un domaine de taille importante [18] dure moins de deux minutes.

4.2.3 Requêtes

Le langage utilisé pour formuler des requêtes sur une base Neo4j est le langage Cypher [25]. Il présente quelques similarités avec SQL. Une requête simple permettant de retourner l'ensemble des nœuds contrôlant le nœud numéro 42 s'exprime de la façon suivante [19] :

```
START adm=node(42) MATCH voisin-->adm RETURN voisin;
```

Il est aussi possible de retourner les chemins entre deux nœuds, c'est-à-dire l'ensemble des nœuds et des relations traversés par le chemin. Dans l'exemple suivant, on retourne l'ensemble des chemins de longueur inférieure ou égale à 3 permettant de contrôler le nœud numéro 42 :

```
START adm=node(42) MATCH path = voisin-[*0..3]->adm RETURN path;
```

18. Par exemple, un des domaines de production étudié comprenait plusieurs millions de relations et plusieurs dizaines de milliers de nœuds.

19. Le numéro d'un nœud est son identifiant unique. La correspondance entre cet *id* et le nom de l'objet associé s'obtient par recherche dans un index de la base de données par exemple.

Les requêtes de cette forme permettent a priori de résoudre le problème de la connectivité du graphe : en choisissant une taille maximale suffisamment importante pour la recherche de chemins, on doit retrouver l'ensemble des nœuds contrôlant le nœud cible. Cependant, cela s'avère impossible en pratique sur des données réelles : le nombre de nœuds et de relations est trop important, et les auteurs n'ont pu obtenir de résultats au-delà d'une longueur maximale de 5 [20]. Or, il existe souvent des chemins de contrôle plus longs. L'approche choisie pour contourner ce problème est d'effectuer la recherche de proche en proche.

1. Choisir un nœud cible $N_{0,0}$.
2. Rechercher l'ensemble des nœuds contrôlant directement $N_{0,0}$. On les note $N_{1,j}$.
3. Pour chaque $N_{1,j}$, on recherche l'ensemble des nœuds le contrôlant directement et n'appartenant pas aux $N_{1,j} \bigcup N_{0,0}$.
4. On itère la procédure pour i allant de 2 à D, distance pour laquelle il n'existe plus de nœud contrôlant un $N_{D,j}$ et n'appartenant pas aux $N_{i,j}$ déjà trouvés.
5. D correspond alors à la longueur maximale des chemins permettant de contrôler $N_{0,0}$.

On peut alors définir le sous-graphe de contrôle comme l'ensemble des nœuds $N_{i,j}$ et des relations r_i mises en jeu lors de transitions vers $N_{0,0}$.

4.3 Visualisation

La méthode présentée dans la partie précédente permet d'extraire le sous-graphe de contrôle d'un nœud arbitraire, c'est-à-dire l'ensemble des nœuds et des relations pouvant être mis en jeu pour contrôler le nœud cible. Se pose alors la question de la visualisation de ce sous-graphe. Neo4j possède une interface Web permettant de faire des requêtes et de visualiser les résultats sous forme graphique. Cependant, cette interface n'est pas utilisable sur des jeux de données aussi importants en volume que ceux mis en jeu lors de l'analyse d'Active Directory complets. La solution retenue

20. Les requêtes naïves ayant une profondeur de 5 ne terminent jamais. Ce problème est sûrement dû à la quantité importante de données, ainsi qu'à la forme particulière de notre graphe : les nœuds privilégiés (tel que celui représentant les administrateurs du domaine) possèdent des relations vers de très nombreux autres nœuds. L'algorithme de parcours s'en trouve perturbé. L'API de Neo4j permet de personnaliser les méthodes de traversée du graphe ; cette fonctionnalité sera probablement utilisée dans une évolution ultérieure de notre outil.

pour simplifier le travail consiste à extraire le sous-graphe et à l'enregistrer au format JSON. La taille du sous-graphe est en général négligeable comparée à celle du graphe complet. Les données sont enfin présentées sous forme d'un SVG grâce à la librairie D3.js [3]. Des graphes extraits et représentés sous cette forme sont présentés dans la partie suivante.

5 Scénarios d'analyse

5.1 Contexte d'analyse

Les analyses présentées ici pourront être effectuées principalement dans deux contextes :

- dans un contexte d'**audit** pour durcissement ou de **réponse à incident** après une compromission. Dans ce cas les différents relevés seront effectués avec des comptes privilégiés, ce qui garantira l'obtention du maximum de relations de contrôle ;
- dans un contexte de **supervision** afin de détecter l'apparition de nouveaux chemins de contrôle. Dans ce cas il est préférable d'utiliser un compte non-privilégié pour effectuer les relevés, la majorité des données permettant d'établir les relations de contrôle étant accessible aux simples utilisateurs. C'est le cas notamment des objets de l'annuaire LDAP ou encore des fichiers de GPO. L'utilisation d'un compte non-privilégié conduit à une analyse certes moins exhaustive, mais qu'il est possible de mener plus fréquemment, en respectant le principe du moindre privilège. Si une relation dangereuse apparaît, une alerte peut être levée.

5.2 Identification des cibles

L'identification des cibles est un point crucial de l'analyse. En effet, étant donné les quantités de données manipulées et les ressources nécessaires pour la génération des chemins, il est nécessaire de cibler un périmètre restreint afin que la tâche soit humainement et techniquement réalisable dans un temps correct. On peut notamment se concentrer sur trois types de cibles privilégiées.

5.2.1 Cibles natives

Comme l'indique la documentation Microsoft [12], il existe par défaut un grand nombre de groupes et utilisateurs privilégiés dans une installation Active Directory. Par défaut, on considèrera notamment :

- les groupes suivants :
  - Enterprise Administrators, Domain Administrators, Schema Administrators, Administrators,
  - Domain Controllers, Enterprise Domain Controllers, Read-only Domain Controllers, Enterprise Read-only Domain Controllers,
  - Account Operators, Server Operators, Backup Operators, Print Operators,
  - Replicators,
  - Group Policy Creator Owners,
  - Cert Publishers ;
- ainsi que les comptes suivants :
  - Administrator,
  - krbtgt,
  - les comptes de contrôleurs de domaine (DC et RODC).

5.2.2 Cibles issues de l'audit des permissions des partages de fichiers

L'objectif des mesures de sécurité qu'une entreprise met en place est d'empêcher un accès illégitime à ses données. Les attaquants ne cherchent pas forcément à obtenir un niveau de privilège absolu mais à obtenir le niveau de privilège suffisant pour pouvoir accéder à un maximum de données en vue de les exfiltrer. Celles-ci sont habituellement stockées sur des partages de fichiers. Un utilisateur « lambda », n'appartenant à aucun groupe d'administration et n'ayant donc pas de privilèges importants au niveau Active Directory peut donc se révéler essentiel s'il lui est permis d'accéder à un gros volume de données sur ces partages.

Il est donc nécessaire d'auditer les partages de fichier afin d'identifier et d'intégrer dans notre analyse :

- les groupes et utilisateurs ayant explicitement accès à une grande quantité de données ;
- les groupes et utilisateurs cumulant les appartenances à de nombreux groupes, cumulant ainsi les accès (les permissions effectives peuvent être plus importantes que les permissions explicites).

Cela pourra être réalisé au travers d'un audit croisant les permissions du système de fichiers des différents partages de fichiers de l'entreprise (majoritairement NTFS) avec la hiérarchie des groupes et utilisateurs.

5.2.3 Cibles représentant un intérêt particulier dans le contexte d'analyse

On pourra également considérer des utilisateurs particuliers en fonction du

contexte. Par exemple s'ils ont accès à d'importants documents, auquel cas ils peuvent faire l'objet d'attaques ciblées. Mais également pour appliquer une logique différente de celle des cas précédents : au lieu d'étudier les chemins de contrôle menant jusqu'aux cibles sélectionnées, il est possible de partir d'une cible précise pour étudier tous les chemins qui en sont issus, afin d'identifier le niveau de privilège maximal atteignable. On pourra ainsi étudier :

- les utilisateurs « VIP » ;
- les cibles suspectes, ou connues comme compromises ;
- les groupes rassemblant un grand nombre de membres connus comme ne devant pas maîtriser d'objets particuliers (`Domain Users` ou `Domain Computers` par exemple).

5.3 Scénarios

Cette section présente des exemples de graphes de chemins de contrôle générés à l'aide de notre outillage. Ils sont tirés de domaines vierges, de domaines de test peuplés artificiellement ou de domaines réels. Les données permettant leur génération ne contiennent pas forcement toutes les relations de contrôle présentées dans cet article. Ainsi ces graphes peuvent être complétés et ne servent ici qu'à illustrer les possibilités de notre méthode d'analyse.

Pour des raisons de lisibilité des graphes, il n'est pas possible d'indiquer en légende le nom de chaque nœud et le type de chaque relation. Nous ferons donc apparaître ces éléments sur des sous-parties plus détaillées de ces graphes après la vue générale lorsque cela s'avère nécessaire.

La légende des graphes est donnée par la figure 13 ci-dessous.

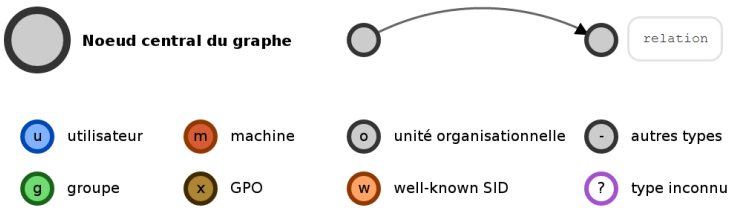


Figure 13. Légende des graphes présentés.

5.3.1 Domaine vierge : chemins par défaut

Cette partie étudie trois domaines « vierges » de niveaux fonctionnels

différents dans leur état post-installation par défaut, auxquels on a simplement rajouté :

- une unité organisationnelle `NewOU` située sous la racine du domaine ;
- un utilisateur `user` sans privilèges particuliers placé dans `NewOU` ;
- un utilisateur `operator` placé dans `NewOU` et membre de tous les groupes d'opérateurs ;
- une machine serveur étant l'unique contrôleur de domaine (Windows Server 2003, 2008R2 et 2012R2) ;
- une machine cliente, dont la version du système d'exploitation correspond à celle du serveur utilisé (Windows XP, 7 et 8.1).

Elle a pour but d'introduire les différents types de graphes qu'il est possible d'obtenir et de produire une situation initiale de référence à laquelle nous pourrons comparer des graphes provenant de domaines plus complexes.

5.3.1.1 Premiers graphes : exhaustivité et lisibilité des graphes générés

Les premiers graphes exhaustifs générés par notre outillage se sont avérés trop complexes et peu lisibles. En effet les groupes privilégiés du domaine (groupes d'administrateurs, d'opérateurs, *well-known SID* `SYSTEM`, etc.) contrôlent chacun énormément d'objets. De ce fait ils sont présents sur quasiment tous les graphes générés et possèdent des relations vers une grande majorité des autres acteurs apparaissant sur le graphe, ce qui le surcharge fortement.

Il est plus judicieux de générer dans un premier temps des graphes contenant uniquement les plus courts chemins par rapport au nœud central du graphe. Ils serviront à l'identification des acteurs, avant de réafficher au besoin l'ensemble des chemins pour certains nœuds identifiés comme anormaux. La figure 14 place côte à côte un graphe complet contenant tous les chemins et un second ne présentant que les plus courts, pour illustrer ce problème de lisibilité. Nous présenterons par la suite majoritairement des graphes faisant intervenir les plus courts chemins.

5.3.1.2 Chemins de contrôle vers un objet

Le premier type de graphe qu'il est possible de générer est un graphe représentant tous les chemins aboutissant à un objet particulier. Cela nous permettra d'identifier tous les utilisateurs et groupes pouvant maîtriser notre objet, mais aussi tous les éléments qui peuvent intervenir dans cette prise de contrôle (machines, GPO, OU, etc.). Rappelons que dans les graphes, les relations sont orientées de l'objet maître vers l'objet esclave.

La figure 15 représente un graphe de contrôle arrivant au groupe `Domain admins` d'un domaine de niveau fonctionnel *Windows Server 2012R2*.

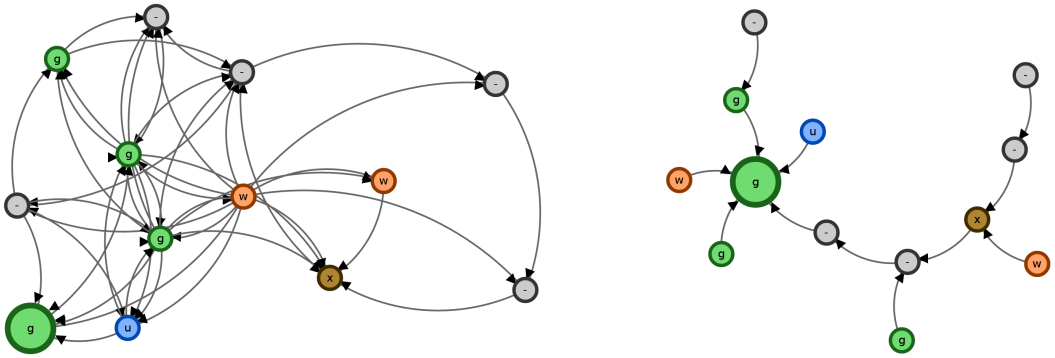


Figure 14. Représentations d'un graphe de contrôle de manière « complète » (à gauche), puis de manière « simplifiée » ne faisant intervenir que les plus courts chemins par rapport au nœud central (à droite).

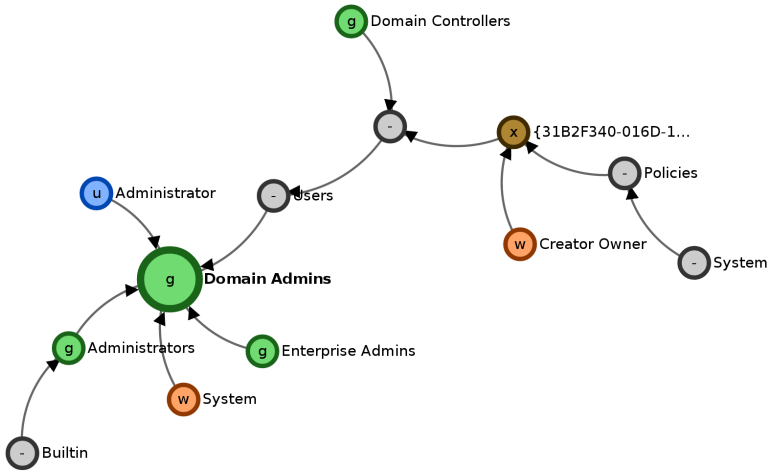


Figure 15. Chemins de contrôle vers le groupe d'administrateurs du domaine (Windows Server 2012R2).

On y retrouve principalement trois types d'éléments :

- les entités très privilégiées du domaine, à savoir les groupes `Administrators`, `Enterprise Admins` et le *well-known SID* `SYSTEM` qui possèdent une ACE de contrôle total, ainsi que le compte `Administrator` appartenant au groupe ;
- la GPO existante par défaut s'appliquant sur tout le domaine (nommée *Default Domain Policy* et possédant toujours le GUID `31B2F340-016D-11D2-945F-00C04FB984F9`) ;
- des éléments de hierarchie pouvant permettre de contrôler les premiers éléments (par exemple les conteneurs formant le chemin `CN=Policies,CN=System,DC=domain`, qui peuvent permettre de po-

sitionner une ACE de contrôle héritée par la *Default Domain Policy* s'ils sont maîtrisés).

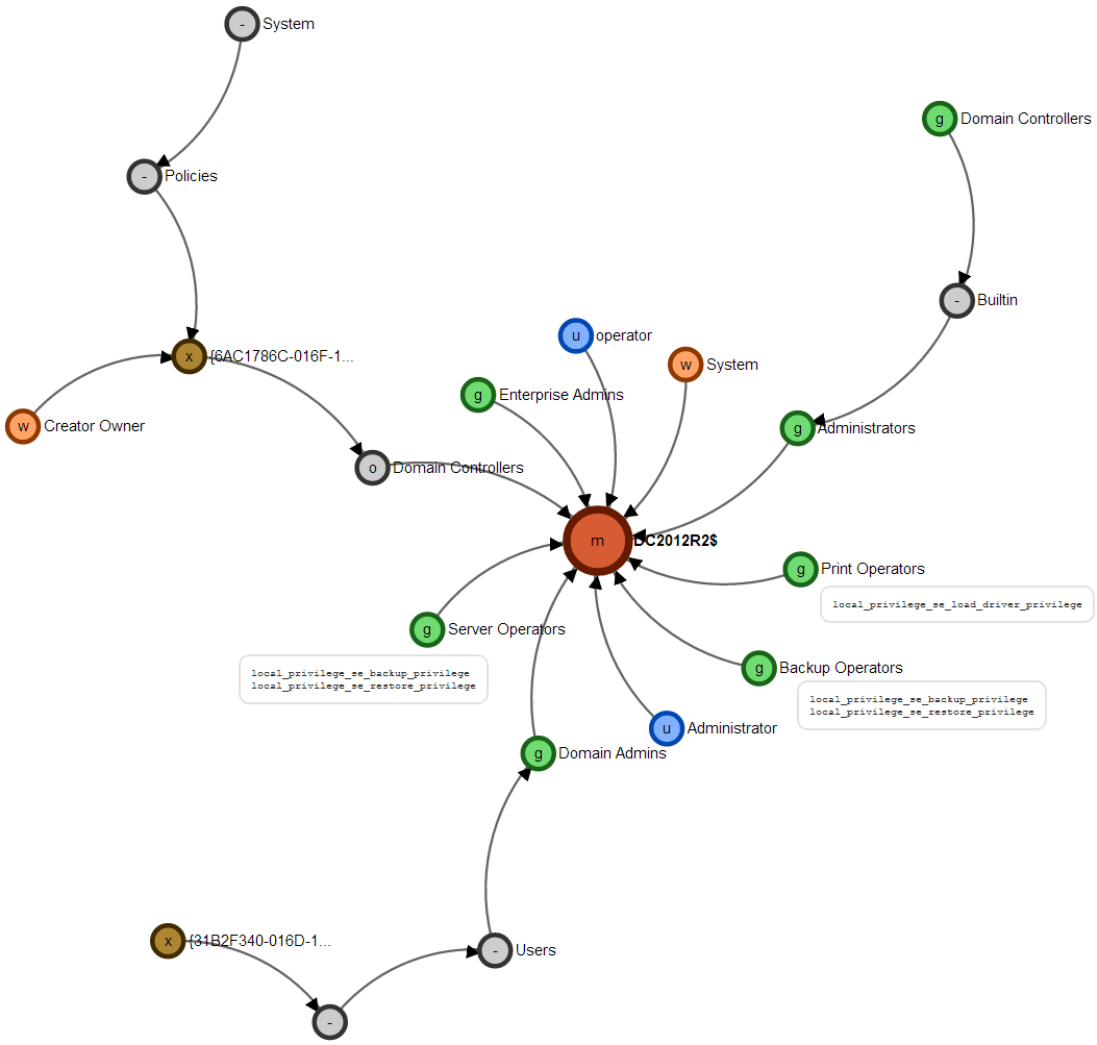


Figure 16. Chemins de contrôle vers un contrôleur de domaine (Windows Server 2012R2).

Il est également possible d'appliquer cette méthode au contrôleur de domaine, comme le montre la figure 16. Ce graphe nous permet entre autres d'identifier plus précisément les privilèges dont disposent certains comptes d'opérateurs :

– comme pressenti, les opérateurs de serveur et de backup disposent des privilèges `SeBackupPrivilege` et `SeRestorePrivilege` leur permettant de lire et écrire n'importe quel fichier du système ;

– de manière moins intuitive, les opérateurs d'impression disposent du privilège `SeLoadDriverPrivilege` leur permettant de charger des modules noyau. Malgré leur nom ordinaire, il s'agit donc d'un groupe privilégié.

On retrouve également sur ce graphe les acteurs privilégiés (`Domain/Enterprise Admins`, `Administrators`, `SYSTEM`) ainsi que les deux GPO existantes par défaut : la *Default Domain Policy* (`{31B2...}`) et la *Default Domain Controllers Policy* (`{6AC1...}`).

5.3.1.3 Chemins de contrôle partant d'un objet

À l'inverse du graphe précédent, il est également possible de générer des graphes représentant tous les chemins provenant d'un objet particulier. Cela nous permettra de représenter l'étendue du pouvoir de cet objet sur le domaine.

Afin d'illustrer cela pour un compte fortement privilégié, la figure 17 représente un graphe de contrôle partant du groupe `Domain admins` d'un domaine de niveau fonctionnel *Windows Server 2012R2*.

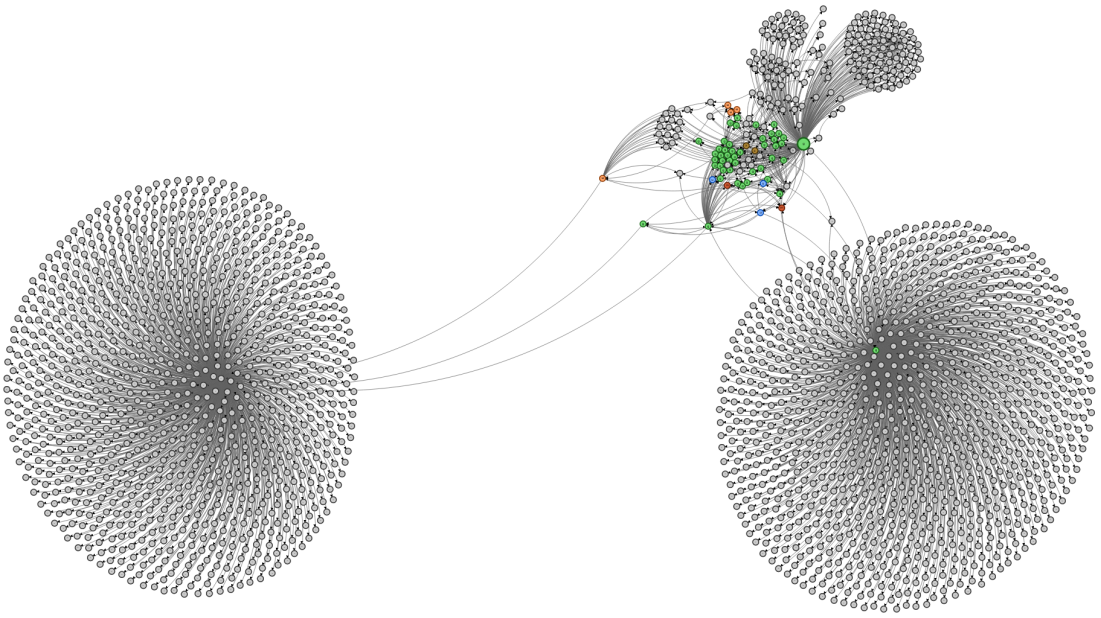


Figure 17. Chemins de contrôle partant du groupe d'administrateurs du domaine, situé au sein de l'amas central (Windows Server 2012R2 ; limite de 1000 nœuds enfants pour chaque parent).

Ce graphe permet d'illustrer de manière explicite :

– le niveau de pouvoir conféré aux administrateurs du domaine, étant donné le nombre important de nœuds représentés (en réalité incom-

plet, car une limite de 1000 nœuds enfants pour chaque parent a été volontairement imposée) ;

- la différence de contrôle direct sur les objets de l'annuaire entre les administrateurs du domaine, les administrateurs de l'entreprise et les administrateurs du schéma, représentée par les trois amas de nœuds visibles :
  - l'amas central « non circulaire » rassemble les objets situés dans la partition de domaine, qui sont contrôlés directement par `Domain Admins` situé en son centre,
  - l'amas situé en dessous rassemble les objets de la partition de configuration, qui eux sont contrôlés directement par `Enterprise Admins` également présent en son centre,
  - l'amas de gauche rassemble quant à lui les objets de la partition de schéma. Ils sont controlés entre autres par `Schema Admins`, mais ce groupe ne se trouve pas directement en son centre, c'est le conteneur `Schema` qui s'y trouve. Cela nous indique qu'à l'inverse des deux premiers groupes d'administrateurs, les objets du schéma ne sont pas directement sous le contrôle des administrateurs du schéma mais *via* l'héritage de permissions positionnées à la racine de la partition.

  Les administrateurs de domaine ne sont donc pas directement maîtres des objets de configuration et de schéma, mais peuvent le devenir par exemple *via* les relations de contrôle suivantes :

```
MATCH m-[rel]->n
WHERE m.name =~ "cn=domain admins,.*"
AND n.name =~ "cn=(enterprise|schema) admins,.*"
RETURN type(rel) AS relation, n.name AS target ;

+-----------------------------------------------------+
| relation                | target                    |
+-----------------------------------------------------+
| ad_owner                | cn=enterprise admins,...  |
| stand_right_write_dac   | cn=enterprise admins,...  |
| stand_right_write_owner | cn=enterprise admins,...  |
| ext_right_all           | cn=enterprise admins,...  |
| write_prop_all          | cn=enterprise admins,...  |
| ad_owner                | cn=schema admins,...      |
| stand_right_write_dac   | cn=schema admins,...      |
| stand_right_write_owner | cn=schema admins,...      |
| ext_right_all           | cn=schema admins,...      |
| write_prop_all          | cn=schema admins,...      |
+-----------------------------------------------------+
```

5.3.1.4 Différences entre domaines de niveaux fonctionnels différents

Un des intérêts à relever les relations de contrôle sur des domaines vierges

de niveaux fonctionnels différents est de pouvoir identifier les modifications des chemins de contrôle ayant eu lieu entre chaque version.

Ainsi, il est possible d'observer un changement majeur dans la gestion des groupes privilégiés et d'opérateurs entre des domaines de niveau 2003 et 2008R2. La figure 18 permet de visualiser ce changement. Elle représente les chemins de contrôle partant du groupe `Account Operators` pour les deux niveaux fonctionnels. Le graphe est radicalement différent entre les deux versions :

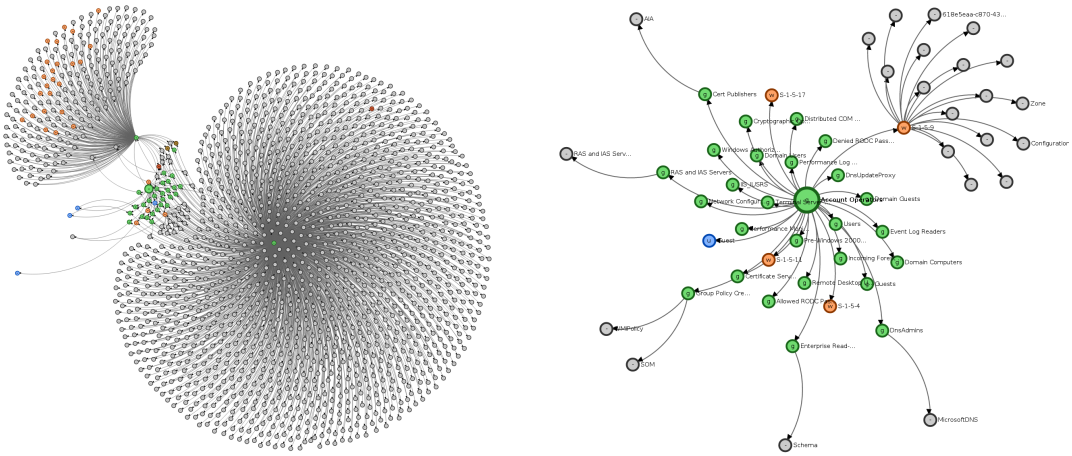


Figure 18. Plus courts chemins de contrôle partant du groupe des opérateurs de compte (Windows Server 2003 à gauche et 2008R2 à droite) (de longueur limite 2).

– pour le niveau 2003, il rappelle le graphe précédent représentant le contrôle total des administrateurs de domaine. En effet on retrouve la quasi-totalité des objets du domaine sous le contrôle des opérateurs de compte, car ces derniers possèdent directement des ACE de contrôle total sur les éléments suivants, en faisant un des groupes les plus privilégiés du domaine :

```
Account Operators
Domain Admins
Domain Controllers
Enterprise Admins
krbtgt
Print Operators
Replicator
Schema Admins
Server Operators
```

– pour le niveau 2008R2, ces relations ont été révisées et ce sont maintenant `Domain/Enterprise Admins`, `Administrators` et `SYSTEM` qui

les possèdent. Cependant le groupe `Account Operators` garde le contrôle de plusieurs groupes d'importance (`Domain Users`, `Domain Computers`, etc.).

De la même manière, nous avons recherché les différences entres les domaines vierges de niveau 2008R2 et 2012R2, mais cette fois sans identifier de changement radical. Les différences existantes font uniquement ressortir l'apparition de nouveaux objets liés aux nouvelles fonctionnalités Active Directory [21], par exemple :

- des objets liés à la virtualisation, comme les groupes `Cloneable Domain Controllers`, `Hyper-V Administrators` ou le droit étendu `DS-Clone-Domain-Controller` ;
- des objets liés aux *claims* et à l'utilisation des ACE conditionnelles, comme la hiérarchie de conteneurs présente dans `CN=Claims Configuration,CN=Services,CN=Configuration,DC=domain` ou les objets de type `ms-DS-Claim-Type`, `ms-DS-Claim-Type` et `ms-Authz-Central-Access-Policy`.

5.3.2 Domaine compromis : identification de portes dérobées

Cette partie présente des graphes provenant d'un domaine de test peuplé artificiellement sur lequel des portes dérobées ont été rajoutées. L'objectif sera de mettre en évidence ces portes dérobées.

Dans un premier temps il est nécessaire d'identifier des acteurs anormaux. Pour cela il est possible d'utiliser des graphes centrés sur les cibles privilégiées définies dans la partie 5.2 et d'inspecter notamment :

- les acteurs ne figurant pas dans les graphes similaires provenant de domaines vierges ;
- les relations « anormales » du point de vue fonctionnel, comme le contrôle d'un objet sur un groupe auquel il n'appartient pas (traduisant une relation d'un autre type) ;
- les acteurs possédant une relation donnée un nombre de fois significativement différent des autres acteurs (par exemple une dizaine, quand tous les autres en ont soit une unique, soit plusieurs centaines).

On retrouve sur le graphe centré sur `Domain Admins` (figure 19) les mêmes entités privilégiées que sur le graphe d'un domaine vierge ainsi que la branche liée à la *Default Domain Policy*. Plusieurs nouvelles branches ont également fait leur apparition. On y trouve certaines anomalies :

21. Les nouvelles relations identifiées sont des relations de propriétaires ou des ACE conférant le contrôle total à des entités déjà privilégiées (principalement administrateurs, administrateurs du domaine et de l'entreprise, opérateurs de compte et `SYSTEM`).

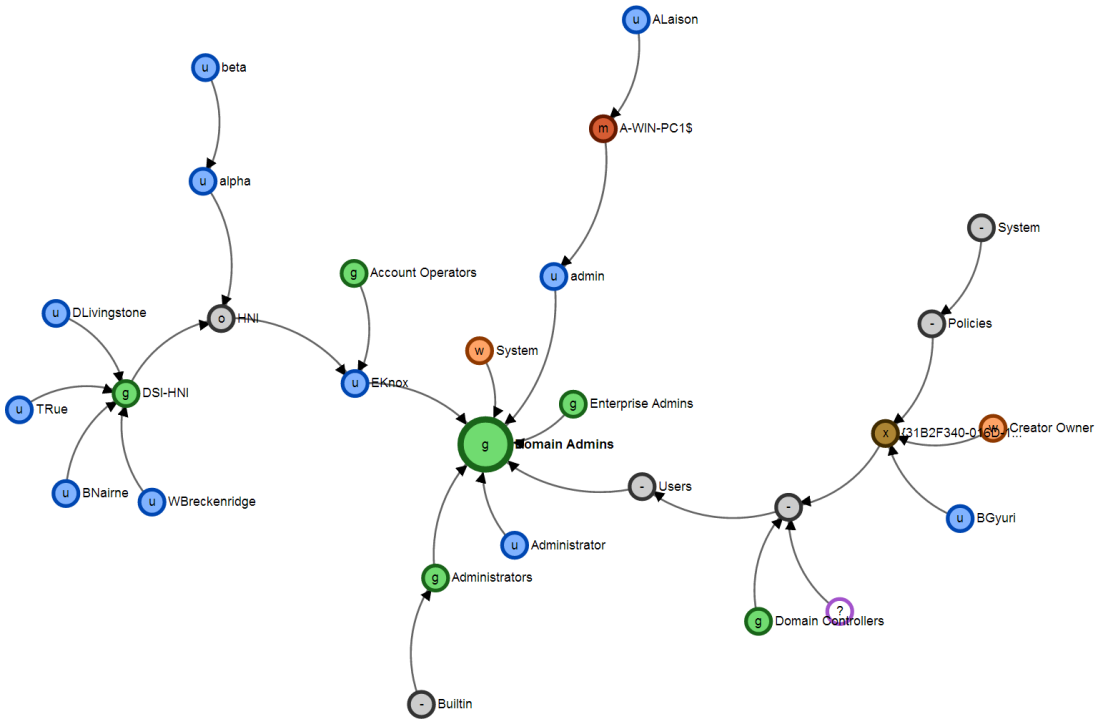
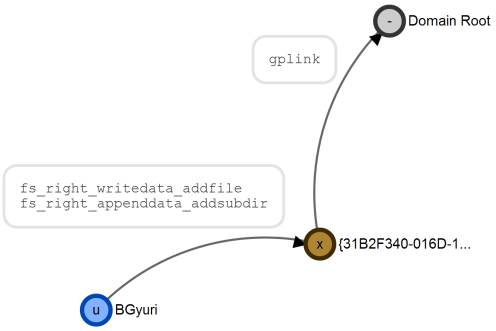


Figure 19. Graphe centré sur le groupe d'administrateurs d'un domaine compromis.

- l'utilisateur `BGyuri` possède des droits de modification des fichiers de la *Default Domain Policy* sur le Sysvol (figure 20) dont il n'est pas propriétaire et pour laquelle il ne possède pas de droits sur l'objet de GPO ;
- l'utilisateur `EKnox` n'est pas membre de `Domain Admins` mais possède des permissions lui conférant son contrôle total. Un graphe des relations immédiates partant de cet utilisateur (figure 21) montre qu'il contrôle toutes les entités les plus privilégiées du domaine, traduisant la modification du descripteur de sécurité de l'`AdminSdHolder` ;
- l'utilisateur `beta` maîtrise un second utilisateur, `alpha` (figure 22). Cette situation est d'autant plus suspecte qu'il ne s'agit pas d'une relation de propriétaire. `Alpha` possède quant à lui un contrôle total sur l'OU `HNI`, qui peut être liée à une délégation d'administration, légitime ou non ;
- l'utilisateur `ALaison` est membre du groupe d'administration local à la machine `A-WIN-PC1$`, or cette machine est utilisée par le compte `admin`, membre de `Domain Admins` (figure 23) ;
- enfin, l'utilisateur `DLivingstone` possède une relation vers le groupe `DSI-HNI`. Cependant, à l'inverse des autres comptes `BNaire`, `TRue` et `WBreckenridge` il ne s'agit pas d'une appartenance au groupe,



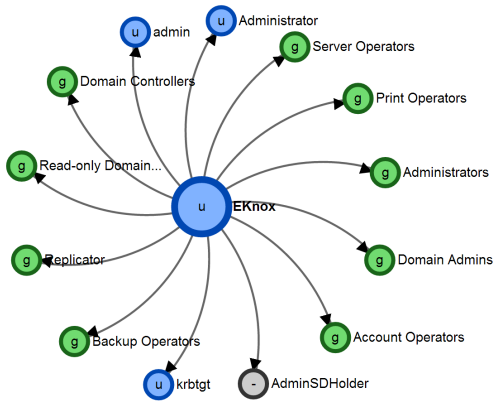
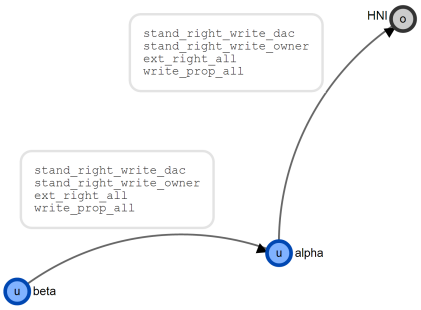


Figure 20. À gauche : utilisateur possédant des droits de modification des fichiers d'une GPO liée à une OU.

Figure 21. À droite : membre possédant une ACE de contrôle total sur tous les objets protégés par l'`AdminSdHolder`.



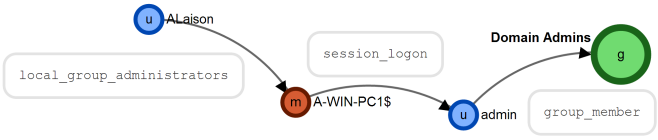


Figure 22. À gauche : contrôle d'un utilisateur sur un second, possédant lui aussi des privilèges sur une OU.

Figure 23. À droite : utilisateur étant administrateur local d'une machine utilisée par un administrateur du domaine.

mais d'une permission d'écriture de toutes les propriétés, ce qui est suspect (figure 24).

Toutes ces situations sont anormales et devront être qualifiées par les auditeurs en partenariat avec l'entité auditée pour juger de leur caractère malveillant. Quoi qu'il en soit, elle devront ensuite être supprimées et remplacées si besoin par des relations plus saines.

5.3.3 Domaine en activité : identification de déviances

Après avoir analysé des domaines de test, cette partie présente des graphes inspirés de domaines réels en activité. Elle a pour but d'illustrer la

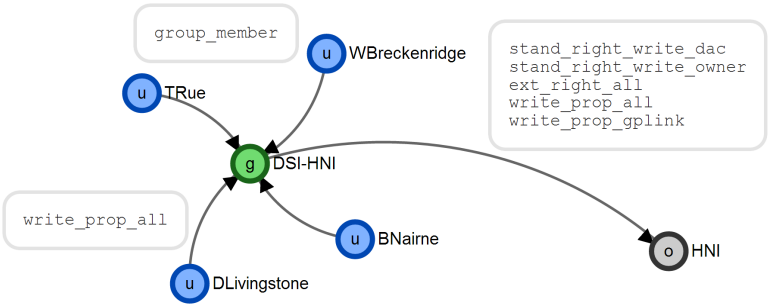


Figure 24. Utilisateur contrôlant un groupe *via* l'écriture de ses propriétés, présent au sein des membres du groupe.

différence de complexité des graphes générés, et d'identifier des déviances dans la gestion du domaine.

Les premiers graphes présentés (figures 25 et 26) représentent respectivement un sous-ensemble des plus courts chemins vers `Domain Admins` et un exemple de hiérarchie de groupes et sous-groupes imbriqués qu'il est possible de rencontrer. Ils mettent en évidence la complexité du domaine.

La première chose que nous pouvons remarquer est l'augmentation du nombre d'acteurs présents par rapport à un domaine vierge. Ainsi, il nous est possible d'identifier de nouveaux groupes privilégiés non-natifs sur le graphe des administrateurs de domaine. Il peut s'agir de groupes d'administration propres à l'entité ou ayant été rajoutés par des applications tierces, comme c'est le cas pour l'application *Exchange* avec le groupe `Exchange Trusted Subsystem` présenté sur la figure 27. Les points notables sont :

- ce groupe possède les droits `WRITE_DAC` et `WRITE_OWNER` sur `Domain Admins` ;
- tous les serveurs Exchange en sont membres, ce qui leur confère également les droits ci-dessus ;
- le compte `Account Operator` en possède le contrôle total, ce qui le replace parmi les maîtres des administrateurs de domaine (relation qu'il avait perdue après le niveau fonctionnel 2003).

Il est également possible de noter des déviances d'administration pouvant impacter l'hygiène du domaine. Par exemple le champ propriétaire de certains objets référence directement des utilisateurs. Ces derniers ont probablement créé ces objets grâce à leur appartenance à des groupes en possédant le droit. Les désigner comme propriétaires peut s'avérer problématique s'ils sont un jour retirés de ces groupes, car ils garderont le contrôle des objets créés. Les permissions peuvent également subsister si le compte est supprimé, ce qui laisse des droits à un SID n'existant plus. Les figures 28 et 29 montrent des exemples d'un compte étant directement

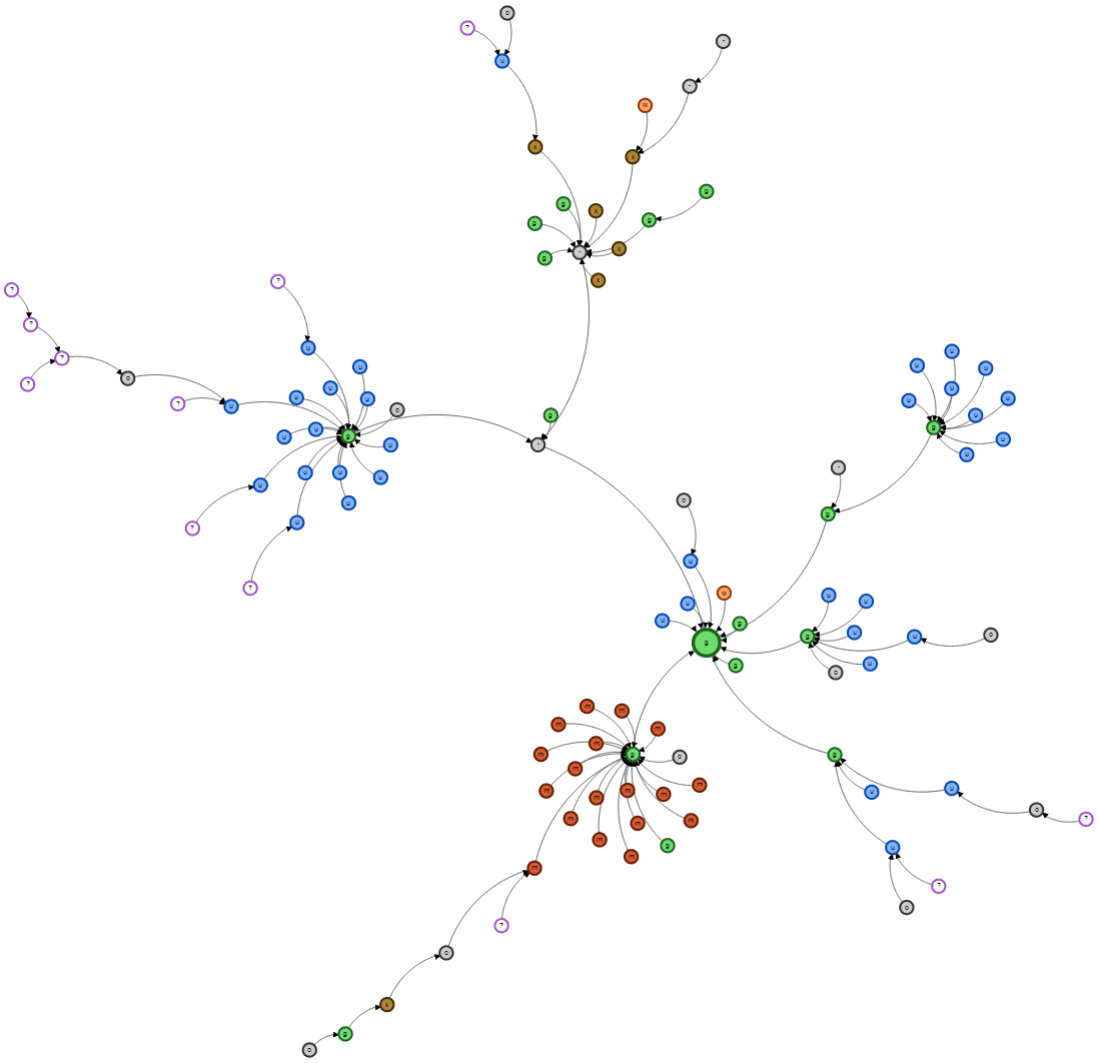


Figure 25. Sous-ensemble des chemins de contrôle vers le groupe d'administrateurs du domaine, d'un domaine en activité.

propriétaires d'une GPO et d'un compte supprimé toujours inscrit comme propriétaire d'un utilisateur.

Enfin, il est aisé de démontrer l'un des avantages principaux de notre méthode d'analyse sur de tels domaines, à savoir la possibilité de générer des chemins de contrôle d'une certaine longueur pouvant chaîner des relations non triviales afin d'identifier les rebonds successifs possibles pour un attaquant. Par exemple la figure 30 représente un chemin de longueur 5, dont les étapes sont les suivantes :

1. le groupe de départ est maître d'une GPO (bien qu'il n'en soit pas propriétaire, il possède notamment les droits en écriture sur toutes les

Figure 26. Illustration d'une hiérarchie complexe de groupes : chaque « îlot » représente un groupe peuplé et la majorité des relations une appartenance à un groupe de niveau supérieur.

propriétés de l'objet de GPO ainsi que les permissions d'ajouter et de modifier les fichiers de GPO) ;

2. cette GPO s'applique sur une OU ;
3. cette OU contient notamment des machines serveurs ;
4. ces machines sont membres d'un groupe ;
5. ce groupe est maître du groupe `Domain Admins` (propriétés `WRITE_DAC` et `WRITE_OWNER`).

Ainsi, le groupe de départ est administrateur du domaine par transitivité, ce qui aurait été bien plus complexe à révéler par une analyse classique. Un attaquant pourrait viser ce groupe plutôt que directement `Domain Admins` en espérant qu'il fasse l'objet d'une attention et d'une sécurisation moindre.

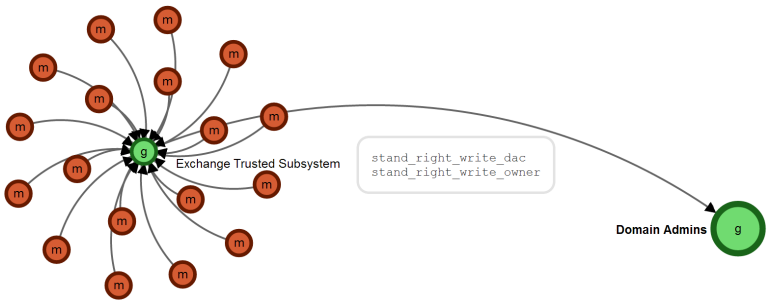
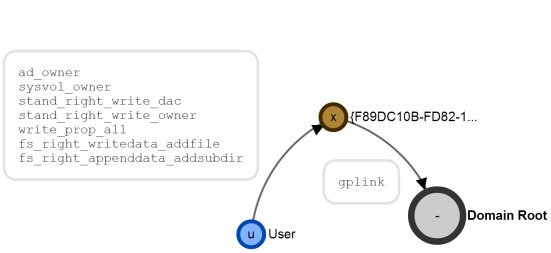


Figure 27. Les serveurs Exchange sont maîtres des administrateurs du domaine *via* leur appartenance au groupe `Exchange Trusted Subsystem`.



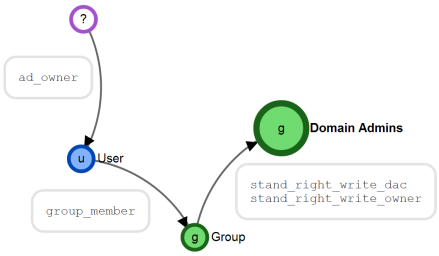


Figure 28. À gauche, un utilisateur directement propriétaire d'une GPO s'appliquant à la racine du domaine (il est directement propriétaire de l'objet ainsi que des fichiers de GPO).

Figure 29. À droite, un utilisateur supprimé toujours propriétaire d'un autre utilisateur pouvant maîtriser les administrateurs du domaine.

6 Conclusion

Cet article présente une méthode d'analyse des environnements Active Directory centrée sur le concept de relations et de chemins de contrôle. Nous avons détaillé la définition, les méthodes de relevé, d'agrégation et d'exploitation de ces relations dans des contextes différents, privilégiés ou non, d'audit et de réponse à incident.

Les avantages d'une telle méthode sont multiples :

- l'exhaustivité des informations agrégées, puisqu'aucune donnée n'est exclue lors des phases de relevé et d'agrégation ;
- l'extensibilité, puisque de nouvelles relations peuvent être ajoutées lors de l'étude de nouveaux objets, propriétés ou droits, introduits par exemple par les nouvelles fonctionnalités des prochaines versions d'Active Directory ;
- l'assistance à la compréhension du domaine, grâce à des représentations graphiques permettant entre autres la mise en évidence de groupes privilégiés non-natifs ;

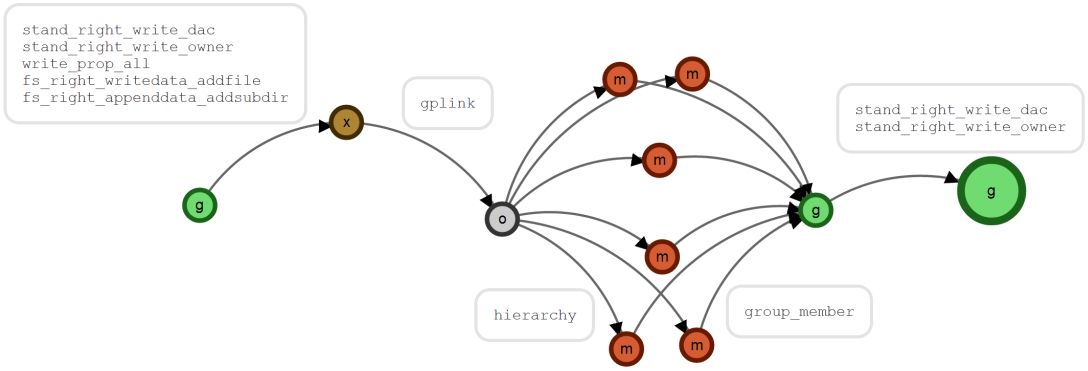


Figure 30. Exemple d'un chemin de longueur 5 chaînant : permissions sur une GPO (objet et fichiers de GPO), application de cette GPO, appartenance à un conteneur, appartenance à un groupe et permissions sur l'objet final, le groupe des administrateurs du domaine.

- le gain de temps lors de traitement d'incidents, grâce à l'identification d'un périmètre atteignable par des comptes particuliers, permettant d'orienter les analyses ;
- la possibilité d'élaborer un plan d'action, consistant en la suppression de relations identifiées comme anormales ;
- l'aide à la décision, grâce à la génération d'informations visuelles permettant de représenter la criticité d'une situation.

Enfin, plusieurs évolutions peuvent être considérées, tant sur la méthode que sur l'outillage. Par exemple :

- modéliser les implications de sécurité des relations d'approbation entre forêts Active Directory pour en déduire de nouvelles relations de contrôle ;
- parfaire la modélisation des mécanismes complexes liés aux conteneurs, comme l'héritage des ACE ou encore l'héritage et l'application des GPO, puisque des particularités comme la propriété `AdminCount`, le bloquage de l'héritage (des ACE et des GPO), les GPO *enforced*, etc. sont complexes à gérer et peuvent introduire des faux positifs lors du relevé des relations ;
- appliquer la méthode à d'autres contextes similaires, par exemple l'audit des permissions d'un système de fichiers NTFS ou une analyse orientée boîtes aux lettres Exchange plutôt que centrée sur les objets de l'annuaire (prenant en compte les descripteurs de sécurité des *mailboxes*, les extensions de schéma et nouveaux droits étendus liés à Exchange et les rôles RBAC) ;
- réaliser des applications adaptées aux différents contextes, telles que :

– un outil d'assistance à la création d'une liste d'actions techniques de durcissement réalisée à partir d'une interface dynamique de présentation des graphes ;
– un outil d'aide à la supervision de la sécurité du domaine, utilisant un compte non-privilégié afin d'être employé de manière périodique, permettant de détecter l'apparition de nouveaux chemins de contrôle.

De telles applications pourront découler aisément de l'outillage actuel réalisant le relevé et la mise en base des relations.

7 Remerciements

Les auteurs tiennent à remercier chaleureusement l'ensemble de leurs collègues pour leur aide précieuse lors de la conception de cette méthode et la rédaction de cet article.

Références

1. Aurélien Bordes. Secrets d'authentification épisode ii - kerberos contre-attaque. https://www.sstic.org/2014/presentation/secrets_dauthentification_pisode_ii__kerberos_contre-attaque/.
2. Peter Clark. Security accounts manager - privilege rights. http://www.beginningtoseethelight.org/ntsecurity/index.htm#4C28668B1B8177C0.
3. D3.js. Data-driven documents. http://d3js.org/.
4. Luc Delsalle. Étude du moteur d'application des stratégies de groupe active directory à des fins d'audit de sécurité. MISC n°73.
5. FlockDB. A distributed, fault-tolerant graph database. https://github.com/twitter/flockdb.
6. Pierre Capillon Géraud de Drouas. Audit des permissions en environnement active directory. https://www.sstic.org/2012/presentation/audit_ace_active_directory/.
7. IETF. Lightweight directory access protocol (v3) - 4.1.12. controls. https://tools.ietf.org/html/rfc2251#section-4.1.12.
8. Microsoft. Active directory technical specification - 3.1.1.4.4 extended access checks. http://msdn.microsoft.com/en-us/library/cc223383.aspx.
9. Microsoft. Active directory technical specification - 5.1.3.3.1 null vs. empty dacls. http://msdn.microsoft.com/en-us/library/cc223515.aspx.
10. Microsoft. Ads_rights_enum enumeration. http://msdn.microsoft.com/en-us/library/aa772285.aspx.
11. Microsoft. Attribute primarygroupid. http://msdn.microsoft.com/en-us/library/cc220723.aspx.
12. Microsoft. Best practices for securing active directory. http://www.microsoft.com/en-us/download/details.aspx?id=38785.

13. Microsoft. Directory replication service (drs) remote protocol - 4.1.10.3.12 issecretattribute. http://msdn.microsoft.com/en-us/library/dd207719.aspx.
14. Microsoft. Dsamain. http://technet.microsoft.com/en-us/library/cc772168.aspx.
15. Microsoft. An introduction to claims. http://msdn.microsoft.com/en-us/library/ff359101.aspx.
16. Microsoft. Ldap_server_sd_flags_oid. http://msdn.microsoft.com/en-us/library/cc223323.aspx.
17. Microsoft. Mandatory integrity control. http://msdn.microsoft.com/en-us/library/windows/desktop/bb648648.aspx.
18. Microsoft. Robocopy. http://technet.microsoft.com/en-us/library/cc733145.aspx.
19. Microsoft. Security descriptor definition language. http://msdn.microsoft.com/en-us/library/windows/desktop/aa379567.aspx.
20. Microsoft. Top class. http://msdn.microsoft.com/en-us/library/ms683975.aspx.
21. Microsoft. Well-known security identifiers in windows operating systems. http://support.microsoft.com/kb/243330.
22. Microsoft. What are domains and forests ? http://technet.microsoft.com/en-us/library/cc759073.aspx.
23. Microsoft. Windows integrity mechanism design. http://msdn.microsoft.com/en-us/library/bb625963.aspx.
24. Neo4j. Batch insertion api. http://docs.neo4j.org/chunked/stable/batchinsert.html.
25. Neo4j. Cypher query language. http://www.neo4j.org/learn/cypher.
26. Neo4j. A highly scalable, robust (fully acid) native graph database. http://www.neo4j.org/.
27. OrientDB. A document graph nosql dbms. http://www.orientechnologies.com/orientdb/.