Kerberos

Chiffrement symétrique vs chiffrement asymétrique

Les chiffrements symétrique et asymétrique sont deux grandes techniques de chiffrement se basant sur des principes différents pour parvenir à une même fin : la transmission sécurisée d’information d’un émetteur vers son destinataire.

Les deux techniques s’opposent dans le fait que pour le chiffrement symétrique, une unique clé de chiffrement sert à chiffrer et déchiffrer un message alors que le chiffrement asymétrique recourt à deux clés : une pour le chiffrement et une autre pour le déchiffrement.

Le chiffrement symétrique impose qu’émetteur et destinataire possèdent la clé de chiffrement. Cette clé servant à la fois au chiffrage et au déchiffrage, il est impératif que chacune des parties la conserve en sécurité (on parle souvent de clé privée ou clé secrète). Le chiffrement symétrique a le désavantage d’imposer que la clé doive être transmise de l’émetteur au destinataire de façon sécurisée.

Pour le chiffrement asymétrique, chaque émetteur possède une paire de clés, une clé dite publique (elle peut être accessible à n’importe qui sans que cela pose de problèmes de sécurité) servant au chiffrement et une clé privée qui doit être conservée en sécurité et servant au déchiffrement. Un message chiffré avec la clé publique d’une personne ne peut être décrypté qu’avec la clé privée de la même personne. Le chiffrage d’une communication par un émetteur se fait donc avec la clé publique de son destinataire, ce dernier déchiffrant ensuite le message à l’aide de sa clé privée. La transmission de la clé publique du destinataire à l’émetteur peut se faire sans contrainte de sécurité contrairement au chiffrement symétrique. L’interception de la clé publique ne pose en effet pas de problème : la clé publique ne sert qu’à chiffrer et est par construction inutile au déchiffrage.

Le protocole Kerberos

Kerberos est un protocole d’authentification suivant le principe AAA (Authentification Authorization Accounting) chargé d’authentifier, d’autoriser et de surveiller les utilisateurs (clients, services, etc.) désirant accéder aux ressources ou aux services d’un réseau. Kerberos repose sur le principe de SSO (Single Sign-On) : avec une unique authentification, l’utilisateur aura accès à tous les services du réseau[[1]](#footnote-1).

Plutôt que d’authentifier chaque utilisateur auprès de chaque service avec une paire identifiant[[2]](#footnote-2)/mot de passe, Kerberos se base sur un tiers de confiance[[3]](#footnote-3) (et un cryptage symétrique) pour gérer l’authentification : le KDC (Key Distribution Center) qui se sépare en deux services :

* L’Authentification Service (AS)
* Le Ticket Granting Service (TGS)

Une fois authentifié par l’AS, l’utilisateur reçoit de celui-ci un ticket spécifique à la session venant d’être ouverte[[4]](#footnote-4) qui lui permettra de s’authentifier de façon transparente (invisible, automatique) auprès des autres services du réseau. Ce ticket est stocké dans le cache des certificats de sécurité de l’ordinateur client. Cette opération d’authentification par le service Kerberos avec réception, décryptage et mise en cache d’un ticket peut être effectuée par le programme de connexion lorsque l’utilisateur se connecte à son poste de travail (l’opération lui est alors transparente) ou alors par l’exécution de la commande kinit.

Il est sous-entendu que chaque service proposé par le réseau sécurisé avec Kerberos doit être compatible avec ce protocole (permettre l’authentification avec un ticket). On dit que de tels services sont kerberisés.

Si on voulait faire un résumé imagé du fonctionnement, on dirait que l’utilisateur s’authentifie auprès de l’AS (le vigile) qui lui donne un ticket (un badge), le TGT (Ticket Granting Ticket). Ce ticket prouve qu’il s’est authentifié. L’utilisateur se rend ensuite auprès du TGS (~service de gestion des tickets) auquel il confirme qu’il s’est bien authentifié grâce à son TGT. L’utilisateur demande l’accès à un service A. S’il a la permission, le TGS lui remet un nouveau ticket (Ticket Service) qui va lui permettre d’accéder au service demandé.

Détails du protocole

1. L’utilisateur s’identifie à l’aide d’une paire identifiant/mot de passe. Le client Kerberos de sa machine dérive alors du mot de passe la clé secrète de l’utilisateur Kutilisateur. Le client Kerberos contacte ensuite l’AS pour identifier l’utilisateur et requérir un Ticket Granting Ticket (TGT). Pour ce faire, le client envoie à l’AS l’identifiant de l’utilisateur et un horodatage (en clair) ainsi qu’un objet chiffré à l’aide de Kutilisateur contenant entre autres un message d’authentification et le même horodatage.
2. L’AS qui gère aussi la base de données des utilisateurs possède le mot de passe de celui cherchant à s’identifier. Elle va ainsi pouvoir générer Kutilisateur et déchiffrer le message d’authentification. Si le message d’authentification correspond au message attendu, c’est qu’il a bien été chiffré avec Kutilisateur : l’utilisateur est authentifié par l’AS[[5]](#footnote-5). L’AS va aussi contrôler l’horodatage afin d’éviter le rejeu de l’authentification[[6]](#footnote-6). Une fois l’utilisateur authentifié, l’AS génère une clé de session KsessionTGS et un TGT qui contient une copie de cette clé de session. Le TGT consiste en une date d’expiration[[7]](#footnote-7), l’identifiant de l’utilisateur et la clé KsessionTGS le tout chiffré à l’aide dela clé secrète du TGS Ktgs (connue de l’AS et du TGS uniquement). Le ticket va permettre à l’utilisateur de demander des Ticket Services (TS) au TGS. Le TGT et la clé KsessionTGS sont envoyés à l’utilisateur, l’ensemble étant crypté par l’AS à l’aide de Kutilisateur afin que ces informations lui soient uniquement accessibles.
3. L’utilisateur déchiffre le message de l’AS et récupère le TGT ainsi que la clé de session KsessionTGS.
4. Quand il veut requérir un service, l’utilisateur va alors faire une demande de TS au TGS en lui envoyant son TGT ainsi qu’un horodatage encrypté avec KsessionTGS pour l’anti-rejeu. Grâce au TGT, le TGS connait l’identité de l’utilisateur, la date d’expiration du ticket et récupère la clé de session qui va lui permettre de déchiffrer l’horodatage. Si l’utilisateur est validé et possède les permissions, le TGS génère une clé de session Ksession et un TS (contenant une copie de cette clé de session) afin de pouvoir accéder au service demandé. Le TS consiste en une date d’expiration, l’identifiant de l’utilisateur et une clé de session Ksession le tout encrypté avec la clé privée du service Kservice connue uniquement du TGS et du service concerné. Le TS est communiqué à l’utilisateur avec un horodatage et la clé de session Ksession le tout encrypté avec KsessionTGS.
5. L’utilisateur déchiffre la réponse du TGS, contrôle l’horodatage et récupère le TS ainsi que la clé de session Ksession. L’utilisateur envoie alors le TS au service désiré ainsi qu’un horodatage encrypté avec Ksession.
6. Le service final déchiffre le TS à l’aide de sa clé Kservice et récupère ainsi l’identifiant de l’utilisateur, la date d’expiration du ticket et la clé de session Ksession qui lui permet de contrôler l’horodatage. Si tout est validé, le service répond à l’utilisateur en lui envoyant un horodatage chiffré à l’aide de Ksession.
7. L’utilisateur déchiffre l’horodatage à l’aide de Ksession et si ce dernier ne laisse pas présumer d’un rejeu, la communication avec le service est finalement ouverte.

Quelques remarques :

* Kerberos utilise le chiffrement symétrique. On a vu à travers l’exemple que les tickets permettent à l’utilisateur de transmettre de façon sécurisée aux services avec lesquels l’utilisateur veut interagir, les clés secrètes (clés de session) dévolues au chiffrement de leurs communications (sauf dans le cas de l’AS où c’est Kutilisateur qui fait office de clé secrète, clé que l’AS possède déjà).
* Le contenu des tickets n’est pas accessible aux utilisateurs car chiffré à l’aide de clés connues de l’AS et du TGS uniquement dans le cas du TGT et du TGS et du service demandé dans le cas du TS.
* Les étapes 1 à 3 correspondent à la commande kinit qui permet d’obtenir et de mettre en cache un TGT.
* Client et fournisseur de service partageant une même clé secrète Ksession, ils peuvent l’utiliser pour sécuriser leur session. Ces clés de session sont en fait des clés intermédiaires qui permettent aux deux parties de ne pas se transmettre leurs clés privées (Kutilisateur et Kservice) : à aucun moment le service ne se trouve en possession de Kutilisateur et inversement.
* Les tickets (TGT ou TS) sont toujours envoyés au client de façon cryptée (avec Kutilisateur ou KsessionTGS) afin d’éviter qu’ils ne soient utilisables par un pirate s’il parvenait à intercepter la communication.

Avantages et inconvénients de Kerberos

Principal avantage :

* Kerberos évite la transmission de messages non-cryptés sur le réseau.

Inconvénients  :

* Tous les services du réseau doivent être compatibles avec le protocole Kerberos (« kerberisables »). Le système peut par exemple être compromis si un utilisateur présent sur le réseau s’authentifie auprès d’un service non-kerberisé en envoyant son mot de passe en clair.
* Toutes les machines doivent être parfaitement synchronisées sinon les horodatages ainsi que la durée de vie des tickets risquent d’être faussés et plus aucune authentification ne sera possible.
* Kerberos introduit un SPOF (Single Point Of Failure) dans le réseau : si le serveur Kerberos tombe, plus aucune authentification n’est possible. Il faut donc s’assurer de la disponibilité permanente du service.
* Si l’AS de Kerberos est compromis, un attaquant peut avoir accès à tous les services du réseau.
* Kerberos ne chiffre que l’authentification et non les données qui sont ensuite transmises lors de la session : les clés de sessions ne chiffrent que les communications du service d’authentification.

Dans la pratique

Le fichier .keytab

Un fichier .keytab est un fichier binaire contenant les paires d’identifiants/clés Kerberos de l’utilisateur (clés dérivées du mot de passe Kerberos de l’utilisateur). On peut utiliser directement le fichier .keytab pour s’identifier sans mot de passe auprès d’un service kerberisé. Si l’utilisateur modifie son mot de passe Kerberos, ses fichiers .keytabs doivent être régénérés.

L’utilité du .keytab est notamment de permettre à des scripts de s’authentifier sans requérir une intervention humaine ou sans utiliser un mot de passe stocké en clair dans un fichier texte. Le fichier .keytab est juste un moyen de stocker ses clés Kerberos dans un fichier local.

Attention aux permissions : n’importe quelle personne ayant les droits en lecture pour un fichier .keytab peut ensuite en utiliser les clés !

La génération d’un fichier .keytab demande simplement qu’un client Kerberos soit installé sur l’ordinateur, on utilise ensuite le binary ktutil lancé depuis une invite de commande après avoir ajouté sa localisation à PATH (/usr/sbin ou /usr/kerberos/sbin).

Remarque : un fichier .keytab n’est pas lié à l’ordinateur qui l’a généré (la preuve : il est souvent délivré par l’administrateur du réseau).

S’authentifier (en lignes de commande) auprès de Kerberos

**Cette opération revient à demander la génération d’un ticket TGT.**

Sans fichier .keytab

On va utiliser la commande kinit. Pour une paire identifiant/mot de passe Kerberos X2017810/<mot de passe> par exemple. On passe à l’invite :

1. kinit X2017810
2. <mot de passe>

Dans ce cas de figure, on suit la procédure décrite plus haut : la clé utilisateur est générée par le client Kerberos à partir du mot de passe, l’utilisateur s’authentifie auprès de l’AS, etc.

Avec fichier .keytab

On utilise kinit –kt <keytab\_file> <identifiant>.

Ex : kinit -kt /etc/security/keytabs/X2017810.service.keytab X2017810

où X2017810.service.keytab est le fichier .keytab.

Autres commandes Kerberos utiles :

* klist : liste le ticket actif
* kdestroy : détruit le ticket actif
* etc.

1. Reposer sur un principe SSO est pratique mais expose le système à la vulnérabilité du protocole d’authentification. [↑](#footnote-ref-1)
2. On parle plus largement en anglais de *principal* (« commettant» en français) notion qui en sécurité informatique désigne n’importe quelle entité pouvant être authentifiée sur un réseau/système informatique. Un *principal* peutêtre un individu, un ordinateur, un service, une entité comme un processus, un thread, etc. [↑](#footnote-ref-2)
3. On ne peut s’authentifier directement auprès du service demandé. Le service n’accepte que les utilisateurs démontrant qu’ils ont été authentifiés par le tiers de confiance choisi (Kerberos), la preuve étant apportée par le ticket présenté au service. [↑](#footnote-ref-3)
4. En général valide une dizaine d’heures de façon à ce qu’en cas de compromission d’un poste, le pirate n’ait accès aux services que pour une courte durée. Un ticket est en fait un ensemble chiffré (déchiffrable uniquement par le service sollicité : le TGS ou un service tiers) rassemblant un timestamp d’expiration, l’identifiant du sujet et une clé utilisée spécifiquement pour les échanges avec le service sollicité. [↑](#footnote-ref-4)
5. On remarque que ni le mot de passe ni la clé secrète de l’utilisateur ne sont échangés (et donc ne peuvent pas être interceptés). [↑](#footnote-ref-5)
6. Une attaque par rejeu (replay) est une attaque réseau dans laquelle une transmission est malicieusement interceptée et répétée par l’attaquant. Il s’agit d’un type d’usurpation d’identité. L’émetteur envoie un message crypté pour s’authentifier, le décryptage du message par le récepteur valide l’authentification. Si le message est intercepté (même crypté) puis répété par l’attaquant, il sera quand même décrypté par le récepteur permettant ainsi à l’attaquant d’usurper l’identité de l’émetteur. Une technique pour se prémunir contre le rejeu est d’utiliser un horodatage. L’émetteur chiffre à l’aide de sa clé secrète la concaténation d’un message et d’un horodatage fournit par son système d’exploitation. Il joint au message crypté le même horodatage en clair. Le destinataire qui possède aussi la clé secrète va pouvoir décrypter le message et s’il juge l’authentification fiable, il va ouvrir une session de communication avec l’émetteur. Si le message est intercepté et rejoué, l’attaquant va se retrouver à utiliser des informations caractéristique d’une session passée, déjà ouverte (ou au moins depuis un temps jugé trop long). Le destinataire refusera alors la communication. De plus, il ne sert à rien pour l’attaquant de changer l’horodatage passé en clair : le destinataire s’apercevra qu’il n’est pas identique à celui issu du déchiffrage du message. [↑](#footnote-ref-6)
7. L’utilisateur va ainsi pouvoir requérir n’importe quel service pour lequel il a une permission pendant la durée de vie du ticket. A expiration, celui-ci doit être renouvelé. [↑](#footnote-ref-7)