Sécurité NetBios

Thierry WONNER, Nicolas JAUFFRET

10 mars 2003

Table des matières

1 Introduction

L'utilisation de Netbios étant intimement lié à celui de SMB, ce rapport traite essantiellement de la sécurité SMB.

Le protocole SMB est l'un des plus utilisé sur les réseaux locaux. Ce rapport présente, en premier lieu, ce que sont les protocoles SMB et CIFS, et leurs fonctionnements, pour finir, sur une mise en évidence de certains problèmes de sécurité relatifs à ces protocoles.

Qu'est ce que SMB/CIFS ?

Selon Microsoft, CIFS fournit une plateforme ouverte d'échange de fichiers et de services d'impression à travers un réseau. Il est basé sur le protocole SMB (Server Message Block) qui est largement utilisé autant sur des ordinateurs personnels que sur des stations de travail.

- Clients SMB disponibles :
  - de Microsoft : Windows 95, Windows for workgroups 3.x, Windows NT,2000 et XP
  - pour Linux : Smblient de Samba, Smbfs pour Linux

- Serveur SMB disponibles :
  - Samba
  - Microsoft Windows for Workgroups 3.x
  - Microsoft Windows 95
  - Microsoft Windows NT
  - The PATHWORKS family of servers de Digital
  - LAN Manager pour OS/2,SCO,etc
  - VisionFS de SCO
  - TotalNET Advanced Server de Syntax
  - Advanced Server pour UNIX de AT&T
  - LAN Server pour OS/2 d'IBM.

2 Etablissement d'une session

Remarque : Le protocole SMB a été développé pour fonctionner sous DOS (avec un processeur Intel), c'est pourquoi les octects sont ordonnés en 'little-endian' contrairement à l'ordre dans les réseaux.

SMB peut fonctionner en surcouche de TCP/IP, NetBeui, le protocole DECnet et IPX/SPX. Il faut utiliser les noms NetBios quand l'implémentation de SMB tourne au dessus de TCP/IP, DECNet ou NetBeui.

Les noms NetBios seront detaillés dans un chapitre suivant, pour l'instant il est juste nécessaire de savoir qu'un nom NetBios identifie un ordinateur sur un reseau Microsoft.

Le développement de SMB a débuté dans les années quatre-vingts, c'est pourquoi il existe un grand nombre de versions du protocole SMB. Cependant, le plus utilisé (sur Windows95, 98, Windows NT, Windows 2000 et XP) est la version NT LM 0.12. Dans ce rapport, on se basera donc sur cette version.

Avant de continuer, il faut savoir qu'un nom de domaine SMB identifie un groupe de ressources (utilisateurs, imprimantes, fichiers, ...) sur un serveur SMB.

Comment un client établit une session SMB avec un serveur ?

Supposons qu'un client veuille accéder à une resource spécifique sur un serveur.

Voir Figure 1 : Etablissement d'une session

1. Pour commencer, le client interroge le serveur pour obtenir une session NetBios. Le client envoit son nom NetBios encodé au serveur SMB. Celui-ci écoute les connections sur le port 139.
   Le serveur recoit le nom NetBios et réponds avec un paquet de session NetBios pour valider la session. Le client entre ensuite dans l'établissement de la session NetBios, c'est à dire l'identification du client sur le serveur SMB.

2. Le client envoit un paquet de requète "SMB negprot" (negprot pour "negociate protocol"). Le client donne une liste des versions du protocole SMB supportées.
   Ensuite le serveur envoit un paquet de réponse "SMB negprot". Ce paquet contient des informations telles que le nom de domaine SMB, le nombre maximum de connection acceptées, les versions de protocole SMB supportées.

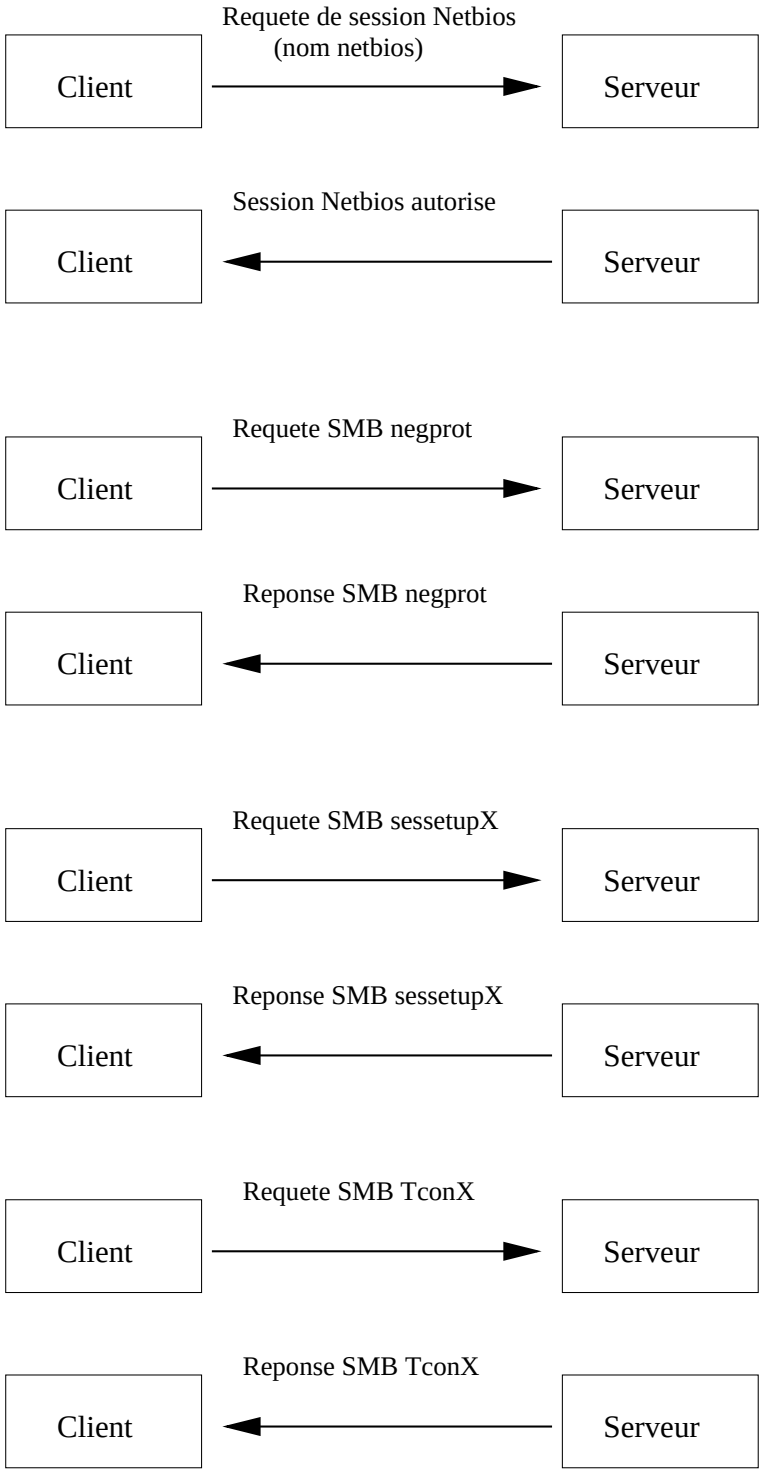


FIG. 1 – Etablissement d'une session

3. Après la négociation de protocole, le client procède à une identification en utilisateur ou en partage ("user identification" ou "share identification") sur le serveur. Ces deux types d'identification seront detaillées dans un chapitre suivant.
   Cette étape est réalisée grâce aux paquets de requète "SesssetupX" (Sesssetup pour "Session Setup and X").
   Le client envoit un couple login/mot de passe ou un simple mot de passe au serveur qui refuse ou autorise la connection avec un paquet de réponse "SesssetupX".

4. Quand le client a terminé avec la négociation et l'identification, il envoit un paquet "tconx" pour spécifier le nom réseau de la ressource qu'il veut acceder.

3 Niveau de sécurité de SMB

Il existe deux types de modèle de sécurité sur SMB :

- Le premier modèle de sécurité est le niveau **partage** (Share Level). Ce modèle de sécurité associe un mot de passe à une ressource partagée du réseau. L'utilisateur se connecte à la ressource (IPC, Disques, Imprimantes) simplement avec un mot de passe. L'utilisateur peut être n'importe qui sur le réseau, il lui suffit de connaître le nom du serveur où se trouve la ressource.
- Le second est le niveau **utilisateur** (User Level). Ce modèle de sécurité est une implémentation ameliorée du premier. Il consiste à associer un couple (login et mot de passe) à une ressource partagée. Ainsi, si une personne souhaite accéder a cette ressource, il doit connaitre le login et mot de passe. Ce modèle de sécurité est utile pour savoir qui fait quoi.

4 Mots de passe

Avec le protocole SMB, lors d'une identification sur un serveur, le mot de passe peut etre envoyé en clair ou en crypté. Si le serveur supporte le cryptage, le client doit pouvoir le gérer. Le serveur connait le mot de passe. Dans le paquet de réponse 'negprot', une clef de cryptage sera envoyée au client ce qui permet à celui-ci de crypter son mot de passe, et de l'envoyer dans le paquet de requete 'Sesssetup'. Le serveur vérifie, ensuitem la validité du mot de passe et autorise ou non la session.

Il faut savoir qu'un mot de passe SMB (non crypté) a pour taille maximum 14 octets. La taille de la clef de cryptage est généralement de 8 octets.
La taille du mot de passe crypté est de 24 octets. Avec un mot de passe ANSI, les caractères du mot de passe sont convertis en majuscule pour le cryptage.

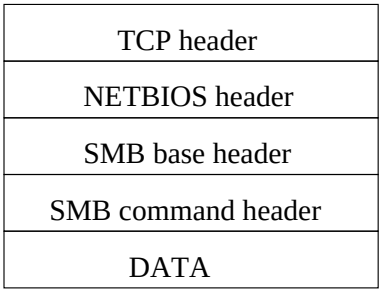


FIG. 2 – Les couches SMB

5 Description de plusieurs paquets SMB

Les paquets les plus importants impliqués dans le protocole SMB seront développés dans cette partie. A chaque type de commande correspond toujours deux types de paquets : les paquets de requête et les paquets de réponse.

5.1 Présentation générale des paquets SMB

Dans la majorité des cas, SMB fonctionne en surcouche du protocole TCP/IP (ce qui sera le cas tout au long de ce rapport). Au dessus de la couche TCP, l'entête NETBIOS (NBT) sera toujours présente. Au dessus de NBT se trouve l'entête de base SMB (SMB base header). Finalement, suit un autre type d'entête, qui dépend des différentes requêtes possible, (SMB Command Header).

Voir Figure 2 : les couches SMB

Explication de la figure :
- Le "SMB Base Header" contient différentes informations, telles que la taille des buffers de réception, le nombre maximum de connections autorisées. Il contient aussi un code qui identifie la commande demandée.
- "SMB Command Header" est un entête avec tout les paramêtres de la commande demandée. (une commande telle que "negociate protocol versions").
- "DATA" sont les données de la commande.

5.2 L'entête de base SMB (SMB Base Header)

Cette entête est utilisée dans tous les paquets SMB, voici sa définition :

```
UCHAR Protocol[4];                  // Contains 0xFF,'SMB'
  UCHAR Command;                      // Command code
  union {
      struct {
          UCHAR ErrorClass;           // Error class
```

```
        UCHAR Reserved;           // Reserved for future use
        USHORT Error;             // Error code
    } DosError;
    ULONG Status;                 // 32-bit error code
} Status;
UCHAR Flags;                      // Flags
USHORT Flags2;                    // More flags
union {
    USHORT Pad[6];                // Ensure section is 12 bytes long
    struct {
        USHORT PidHigh;           // High part of PID
        ULONG  Unused;            // Not used
        ULONG  Unused2;
} Extra;
};
USHORT Tid;                       // Tree identifier
USHORT Pid;                       // Caller's process id
USHORT Uid;                       // Unauthenticated user id
USHORT Mid;                       // multiplex id
UCHAR  WordCount;                 // Count of parameter words
USHORT ParameterWords[ WordCount ];    // The parameter words
USHORT ByteCount;                 // Count of bytes
UCHAR  Buffer[ ByteCount ];       // The bytes
```

- Le champ "protocol" contient le nom du protocole SMB, précédé d'un 0xFF
- Le champ "command" contient la valeur de la commande de requête. Par exemple 0x72 est la commande de négociation de protocole (negociate protocol).
- Le champ "Tid" est utilisé quand le client a réussi à se connecter à une ressource sur le serveur SMB. Le nombre contenu dans le "Tid" identifie cette ressource.
- Le champ "Pid" est utilisé quand le client a réussi à créer un processus sur le serveur. Le nombre contenu dans le "Pid" identifie ce processus.
- Le champ "Uid" est utilisé quand le client a réussi à s'authentifier sur le serveur. Le nombre contenu dans le "Uid" identifie l'utilisateur.
- Le champ "Mid" est utilisé avec le champ "Pid" quand le client a lancé plusieurs requêtes sur le serveur.
- Le champ "Flags2" est aussi important, quand le bit 15 est initialisé, les chaines de caractêres sont encodées en UNICODE.

6 Les noms NetBios

NETBIOS (pour NETwork Basic Input and Output System) est très utilisé dans les réseaux Microsoft. C'est une interface logicielle et un système de nommage des ressources sur un réseau. Chaque ordinateur possède un nom NETBIOS, qui est composé de quinze caractères et d'un seizième qui est utilisé pour identifier le type d'ordinateur (Serveur de nom de domaine, simple machine de bureau, ...).

6.1 Les noms Netbios et leur encodage

Un nom NetBios encodé est composé de 32 octets. Il est toujours donné en lettres majuscules. L'encodage des noms NetBios est relativement simple, l'exemple suivant permet de mieux le comprendre. Il se base sur un ordinateur possédant comme nom NetBios "BILL". Lorsque ce nom est plus petit que 15 caractères, il lui est rajouté un nombre d'espaces necéssaire.
"BILL "
Dans l'exemple, le seizième caractère est 0x00 (voir l'explication dans la section suivante).
Soit en hexadécimal : **0x42 0x49 0x4c 0x4c 0x20 0x20 ..... 0x00**

Chaque octet est coupé en moitié de 4 bits :
0x4 0x2 0x4 0x9 0x4 0xc 0x4 0xc 0x2 0x0 ....

Finalement à chaque groupe de 4 bits, est rajouté le code ASCII de la lettre **'A'** (0x41) :

$$0x4 + 0x41 = 0x45 => valeur ASCII = E$$
$$0x2 + 0x41 = 0x43 => valeur ASCII = C$$

Le nom NetBios obtenu est bien composé de 32 octets.

6.2 Le seizième caractère

Le seizième caractère d'un nom NetBios est un caractère caché. C'est un nombre hexadécimal qui caractérise le nom NetBios.

Voici les principales valeurs que peut prendre ce seizième caractère :
Les noms uniques :

- machine<00> Service Station de travail
- nom_de_domaine<1B> Cette machine est Maitre Explorateur du Domaine
- machine<03> Service Messagerie (net send)
- nom_de_domaine<1D> Maitre Explorateur du réseau ou serveur WINS
- machine<06> Serveur d'accès distant (RAS)
- machine<1F> Service NetDDE
- machine<20> Client WINS
- machine<21> Client RAS

- machine<BE> Agent du moniteur réseau (SMS)
- machine<BF> Serveur SMS
- utilisateur<03> utilisateur connecté localement

Les noms de groupes :

- nom_de_domaine<00> Nom du groupe d'ordinateurs auquel la machine appartient (groupe de travail ou domaine)
- nom_de_domaine<1C> Cette machine est un des controleurs du domaine.
- nom_de_domaine<1E> Cette machine est éligible au rang de maitre explorateur.
- .._ _MSBROWSE_ _.<01> Utilisé par le maitre explorateur pour dire qu'il est LE maitre explorateur sur le réseau.

7 Description des commandes SMB les plus importantes :

Ci-après suivent les commandes nécessaires utilisées par un client désirant accéder à un fichier sur un partage SMB détenu par un serveur :

SMB_COM_NEGOTIATE : doit être le premier message envoyé par le client au serveur. Ceci inclut une liste des protocoles SMB supportés par le client. La réponse du serveur précise quel protocole SMB utiliser.

SMB_COM_SESSION_SETUP_ANDX : fournit les noms et informations nécessaires au serveur pour vérification. La réponse du serveur y inclut l'Uid (clef unique de session pour le client qui servira tout au long de la transaction).

SMB_COM_TREE_CONNECT_ANDX : fournit le nom du partage de disque que le client désire accéder (les imprimantes et InterProcess Communication IPC passent par un autre mécanisme). Une réponse positive du serveur précise le champs Tid (dans le header SMB) qui sera utilisé pour se referrer à cette ressource.

SMB_COM_OPEN_ANDX : transmet le nom de fichier (relatif à la ressource ouverte précisée par le Tid) que le client souhaite ouvrir. Le serveur transmet si la réponse est positive un file ID (Fid) que le client utilisera pour toutes les opérations sur ce fichier.

SMB_COM_READ : le client fournit : Tid, Fid, l'Offset du fichier et le nombre d'octet qu'il désire lire. Le serveur lui renvoie les données correspondantes.

SMB_COM_CLOSE : le client ferme le fichier représenté par Tid et Fid. Le serveur lui renvoie une réponse positive.

SMB_COM_TREE_DISCONNECT : le client se déconnecte de la ressource représenté par Tid.

Rentrons un peu plus en détail pour les étapes suivantes :

7.1 Le protocole de négotiation SMB : negprot

Il est utilisé dans les premières étapes de l'établissement d'une session SMB. Le code de commande correspondant du champs "Command" du header de base SMB vaut : 0x72.
Voici la description des entêtes de requêtes et de réponses de “negprot” :

Request header

```
UCHAR WordCount;            Count of parameter words = 0
```

```
USHORT ByteCount;            Count of data bytes
struct {
  UCHAR BufferFormat;        0x02 -- Dialect
  UCHAR DialectName[];       ASCII null-terminated string
} Dialects[];
```

Ce paquet est envoyé par le client pour donner au serveur la liste des versions de protocoles SMB qu'il supporte.

"Wordcount" : vaut toujours 0.
"ByteCount" : vaut la taille de la structure "Dialects"
"BufferFormat" : vaut toujours 0x02
"DialectName" : est une chaine de charactère contenant le nom des protocoles SMB supportés par le client.

Reply header

```
UCHAR WordCount;           Count of parameter words = 17
USHORT DialectIndex;       Index of selected dialect
UCHAR SecurityMode;        Security mode:
                           bit 0: 0 = share, 1 = user
                           bit 1: 1 = encrypt passwords
USHORT MaxMpxCount;        Max pending multiplexed requests
USHORT MaxNumberVcs;       Max VCs between client and serveur
ULONG MaxBufferSize;       Max transmit buffer size
ULONG MaxRawSize;          Maximum raw buffer size
ULONG SessionKey;          Unique token identifying this session
ULONG Capabilities;        serveur capabilities
ULONG SystemTimeLow;       System (UTC) time of the serveur (low).
ULONG SystemTimeHigh;      System (UTC) time of the serveur (high).
USHORT serveurTimeZone;     Time zone of serveur (min from UTC)
UCHAR EncryptionKeyLength; Length of encryption key.
USHORT ByteCount;          Count of data bytes
UCHAR EncryptionKey[];     The challenge encryption key
UCHAR OemDomainName[];     The name of the domain (in OEM chars)
```

Ce paquet est envoyé par le serveur pour donner au client une liste de protocoles SMB qu'il supporte, le nom du domaine SMB du serveur et la clef d'encodage si necessité il y a.

IMPORTANT :

Le premier champs intéressant est l'octet "SecurityMode". Si le bit 0 est à 1, le niveau de sécurité est au niveau user, sinon, nous avons un niveau de securité au niveau partage. Si le bit 1 est à 1, le mot de passe est encrypté par un DES en mode block.

Le champs "SessionKey" est utilisé pour identifier la session. Il n'y a qu'une seule clef de session pour une session.

Le champs "Capabilities" indique si le serveur supporte les chaines UNICODE, ou bien des commandes NT LM 0.12 particulières.

Les données se trouvent à la fin de l'entête. Elles correspondent aux chaines "EncryptionKey" et "OemDomainName". Leur taille cumulée est donnée par la champ "Bytecount".
La longueur de "EncryptionKey" est donnée par "EncryptionKeyLenght. La longueur de "OemDomainName" est donnée par l'opération "Bytecount"-"EncryptionKeyLenght". "OemDomainName" nous donne le nom de domaine SMB du serveur.

7.2 Session setup and X

Les paquets "Session setup and X" (SessetupX ou bien setupx en abrégeant) sont utilisés pour fournir l'identité d'un utilisateur devant fournir un mot de passe lorsqu'il désire accéder à une ressource.
Le code de commande du champs "Command" du header de base SMB pour le "Session Setup and X" vaut : 0x73.

Request header

```
UCHAR WordCount;                 Count of parameter words = 13
UCHAR AndXCommand;               Secondary (X) command;  0xFF = none
UCHAR AndXReserved;              Reserved (must be 0)
USHORT AndXOffset;               Offset to next command WordCount
USHORT MaxBufferSize;            Client's maximum buffer size
USHORT MaxMpxCount;              Actual maximum multiplexed pending
                                 requests
USHORT VcNumber;                 0 = first (only), nonzero=additional
                                 VC number
ULONG SessionKey;                Session key (valid iff VcNumber != 0)
USHORT                           Account password size, ANSI
CaseInsensitivePasswordLength;
USHORT                           Account password size, Unicode
CaseSensitivePasswordLength;
ULONG Reserved;                  must be 0
ULONG Capabilities;              Client capabilities
USHORT ByteCount;                Count of data bytes;    min = 0
UCHAR                            Account Password, ANSI
CaseInsensitivePassword[];
UCHAR CaseSensitivePassword[];   Account Password, Unicode
STRING AccountName[];            Account Name, Unicode
STRING PrimaryDomain[];          Client's primary domain, Unicode
STRING NativeOS[];               Client's native operating system,
                                 Unicode
STRING NativeLanMan[];           Client's native LAN Manager type,
                                 Unicode
```

Ce paquet donne de nombreuses informations sur le système du client.

Le champs "MaxBufferSize" est très important, il donne la quantité maximum de données que le client peut recevoir. Si il est mis à la valeur 0, le client ne recevra aucune donnée en provenance du serveur.

Les champs "CaseSensitivePassword" et "CaseInsensitivePassword" permettent de savoir si le password sera codé respectivement en caractères UNICODE ou bien en ANSI. L'un des deux champs sera utilisé en fonction de la capacité du serveur à gérer les chaines UNICODE (voir le paquet de réponse du negprot).

Le nombre d'octet des données est fournit par le champs "Bytecount".

Reply header

```
UCHAR WordCount;                Count of parameter words = 3
UCHAR AndXCommand;              Secondary (X) command;  0xFF =
                                none
UCHAR AndXReserved;             Reserved (must be 0)
USHORT AndXOffset;              Offset to next command WordCount
USHORT Action;                  Request mode:
                                bit0 = logged in as GUEST
USHORT ByteCount;               Count of data bytes
STRING NativeOS[];              serveur's native operating system
STRING NativeLanMan[];          serveur's native LAN Manager type
STRING PrimaryDomain[];         serveur's primary domain
```

De nombreuses informations sont fournies par ce paquet : le type de système d'exploitation, la version du serveur SMB, le nom de domaine SMB.

Si la connection ne s'est pas bien passée, les chaines de "NativeOS", "NativeLanman" et "PrimaryDomain" sont vides.

8 Comment retrouver un mot de passe SMB en clair à partir du réseau alors qu'il devrait etre encrypté

Lors de l'établissement de la session, le mot de passe est envoyé au serveur pendant la session SMB setupx. le paquet de retour du SMB negprot contient un bit dans le champs "SecurityMode" qui permet l'encodage ou non du mot de passe.

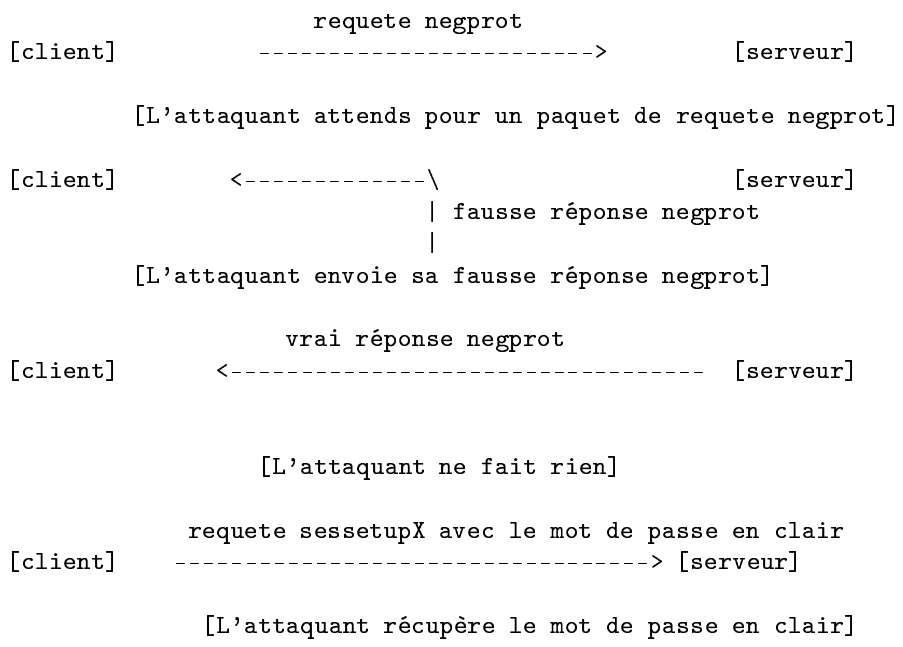
Donc il y a deux possibilités pour obtenir le mot de passe en clair :

La première consistant à obtenir le mot de passe encrypté et de le brute forcé, mais cela peut s'avèrer plutot long...

Des programmes comme LophtCrack (avec SMBGrinder), dsniff ou bien readsmb2 font de l'écoute sur le réseau pour obtenir des mots de passe encryptés.

La seconde possibilités est de détourner la connexion et faire croire au client que le password ne devrait pas être encodé.

Si le serveur est configuré pour encoder les mots de passe, le paquet de retour SMB negprot possède le bit 1 du champs "SecurityMode" à 1. Mais si un attaquant envoie un paquet negprot avec ce bit à 0 avant le serveur, le mot de passe circulera en clair dans le paquet de requête SessetupX. Cela équivaut à créer une race condition.



Ce petit schéma illustre une introduction directe d'un paquet sur le réseau. Dans la plupart des cas, cette méthode ne fonctionne pas car il existe une race condition entre le faux et le vrai paquet de réponse negprot. Il y a aussi d'autre problèmes dus à l'utilisation des switchs, aux sessions qui ne fonctionneront pas à cause des attentes du serveur d'un mot de passe crypté et de la réception de celui-ci d'un mot de passe en clair... Ces problèmes peuvent être évités en utilisant la technique de l'ARP-Poisoining.

Pour résumer, l'ARP-Poisoining permet à un attaquant de rediriger et modifier le traffic entre un client et un serveur.

Dans cette situation, l'attaquant se retrouve entre le client et le serveur, il est le "Man in the middle".

9 L'attaque du "Man in the middle" (l'invisible troisième homme)

Cette attaque permet de passer outre les switches, et récuperer le mot de passe en clair.

Une fois que le traffic est redirigé par l'attaquant (à l'aide de l'ARP Poisoning), supposons que le client émette une requête SMB vers le serveur. Il va envoyer ses paquets sur le port SMB (139) du serveur. L'attaquant va les recevoir mais ne les redirigera pas vers le serveur. Tout le traffic en direction du serveur sera redirigé sur un port local de l'attaquant (à l'aide de NAT et d'iptables). A l'écoute de ce port, un programme de type proxy transparent fera les modifications nécessaires et redirigera les paquets SMB.

```
(Pour info :
#iptables -t nat -A PREROUTING -i eth0 -p tcp -s <CLIENT IP ADDRESS> \
--dport 139 -j REDIRECT --to-port 1139

permet de rediriger les connections du port 139 vers le port 1139.

Pour rediriger l'intégralité du traffic:
#iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
)
```

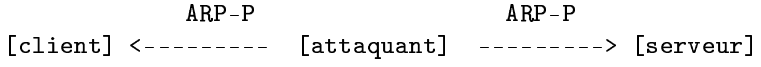
L'attaquant modifiera la réponse negprot pour que le mot de passe circule en clair sur le réseau. L'attaquant récupère aussi la clef d'encryptage. Il met à 0 la taille de la clef d'encryptage et met le bit du champs "SecurityMode" à 0. De cette manière, le mot de passe ne sera pas encrypté.

Le client enverra son mot de passe en clair dans sa requête sessetupx. Quand l'attaquant aura le mot de passe, il l'encrypte avec la clef d'encryption récupérée and envoie la nouvelle requête sessetupx formée au serveur.

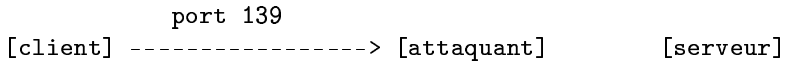
Le serveur, quant à lui, accepte ou refuse la session. A partir de quoi, l'attaquant redirige la réponse du serveur ainsi que tout le traffic à nouveau.

La session fonctionne parfaitement et personne n'aura détecté notre attaquant.

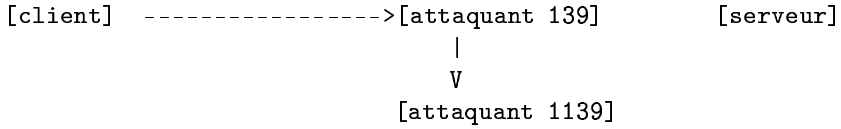
Description :



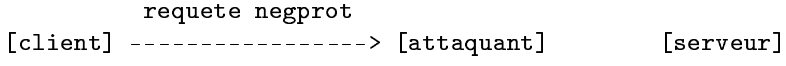
L'attaquant met en place un ARP Poisoining pour rediriger l'intégralité du traffic entre les deux machines.



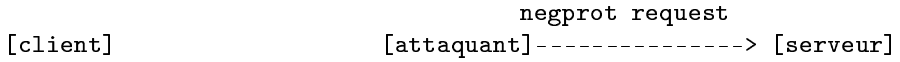
L'attaquant reçoit le premier paquet sur le port SMB standard.



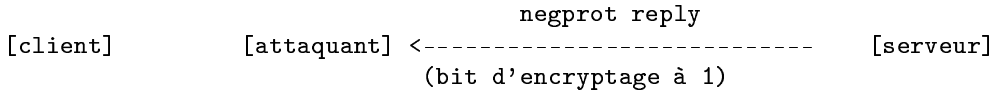
Il redirige ce paquet sur le port local 1139. Sur ce port, un proxy est en écoute.



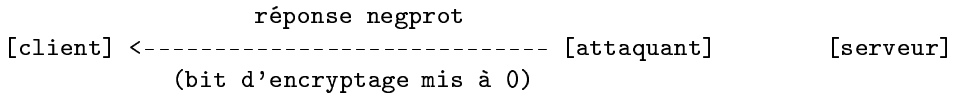
L'attaquant reçoit la requête negprot.



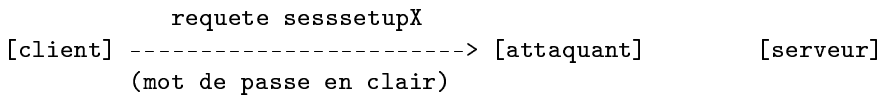
Il la redirige vers le serveur.



Le serveur réponds avec une réponse negprot dont le bit d'encryptage est mis pour recevoir des mots de passes encryptés. L'attaquant ne redirige pas ce paquet, il change d'abord le bit d'encryptage pour recevoir du client des mots de passe en clair.



L'attaquant envoie la réponse modifiée.



Le client envoie le mot de passe en clair, l'attaquant le recoit.

```
                                    requete sesssetupX
[client]              [attaquant] ---------------------->  [serveur]
                                  (mot de passe encrypté)
```

L'attaquant envoie une requête sessetupx au serveur en ayant encrypté le mot de passe.

```
                                    réponse sesssetupX
[client] <-------------- [attaquant] <-----------------  [serveur]
```

Le serveur renvoie une réponse directement redirigée par l'attaquant.

```
[client] <------------> [attaquant] <--------------> [serveur]
```

L'attaquant ne fait plus que rediriger le traffic entre les deux machines et ceux jusqu'à la fin de la session SMB.

10 Conclusion

En utilisant les methodes presentées, la récuperation des mots de passe SMB en clair sur le réseau (alors qu'ils devraient s'y trouver cryptés) devient donc possible.

Toutefois ce rapport ne traitant qu'une partie de la sécurité Netbios (celle liée à l'utilisation du protocole SMB en surcouche de TCP), l'existence d'autres failles de ce protocole est probable.