Sécurité de RDP

Aurélien Bordes, Arnaud Ebalard et Raphaël Rigo
prenom.nom@ssi.gouv.fr

ANSSI

Résumé Cet article présente une étude de sécurité de RDP (*Remote Desktop Protocol*). Il tire son origine du constat effectué sur le terrain que le protocole est très largement (mal) utilisé pour l'administration à distance de systèmes Windows.

Après une revue du besoin et des propriétés attendues, une présentation du protocole est réalisée, suivie d'un historique orienté sécurité.

Les mécanismes de sécurité mis en œuvre sont ensuite analysés en détails et leurs faiblesses résiduelles décrites. En particulier, les modes historiques de protections offerts par RDP, propriétaires, sont particulièrement vulnérables et permettent à un attaquant d'attenter à l'intégrité et à la confidentialité des échanges.

La réalité des implémentations client et serveur Microsoft est ensuite détaillée. Tous ces éléments sont enfin pris en compte dans un ensemble de recommandations et de bonnes pratiques d'utilisation de RDP visant à renforcer la sécurité de l'administration à distance de systèmes Windows.

1 Introduction

1.1 Périmètre de la discussion

D'une simple solution de déport d'affichage dans Windows NT 4.0 TSE, l'écosystème RDP a évolué au fil des années, initialement pour supporter un plus grand nombre de périphériques et de fonctionnalités. Aujourd'hui, le protocole est au cœur d'une architecture de services complexe [5] [1], nécessitant dans certains cas plus d'une demi-douzaine de rôles serveur différents.

L'analyse de sécurité présentée dans cet article se concentre sur le cœur du protocole permettant la protection des données échangées et vise principalement les utilisations classiques de RDP, pour la connexion à distance ou pour des besoins d'administration. Elle ne couvre ainsi pas les évolutions d'architectures récentes (supportées par Windows 2008 R2), le déploiement de RemoteApp, etc.

Du point de vue des implémentations, l'analyse se concentre principalement sur les solutions serveur et client Microsoft, intégrées au système d'exploitation.

1. Une imprimante A2 est nécessaire pour l'impression de ce poster.

1.2 Besoin d'administration à distance

La plate-forme Windows, de par son utilisation massive et historiquement obligatoire des interfaces graphiques, rend nécessaire l'utilisation d'un protocole de déport d'affichage pour de nombreuses tâches d'administration à distance. En effet, même si de nombreux services Windows peuvent maintenant être administrés *via* l'utilisation de RPC[2] depuis un système distant, cette technique n'est pas toujours applicable, notamment pour les systèmes hébergeant des applications non Microsoft. Pour satisfaire ce besoin de déport d'affichage, de nombreuses solutions existent, aux caractéristiques très différentes :

- **VNC** : cette solution historique reste encore très utilisée malgré une sécurité des plus limitée dans son implémentation usuelle. Certaines versions proposent néanmoins des fonctionnalités de sécurité avancées (TLS, authentification Windows, etc.) ;
- **RDP** : il s'agit de la solution développée par Microsoft, extrêmement utilisée, objet de l'étude ;
- **divers produits propriétaires (DameWare, PC anywhere, NetOp, etc.)** : la sécurité de ces solutions varie mais est en général peu étudiée.

1.3 Propriétés de sécurité attendues

Cette sous-section rappelle les principales propriétés de sécurité attendues d'un protocole d'administration à distance. La majorité de celles-ci pourront s'apparenter pour le lecteur à du simple bon sens ; malgré tout, nous verrons dans la suite de l'étude que la plupart des versions de RDP ne satisfont pas ces propriétés.

Authentification La première propriété attendue concerne l'intégration de mécanismes d'authentification sûrs. Ceux-ci doivent notamment être conçus de manière à garantir le caractère mutuel de l'authentification entre l'administrateur et le système visé. Mais il doivent également prévenir les possibilités de rejeu, les attaques par le milieu ainsi que les attaques hors-ligne visant à remonter aux secrets d'authentification client ou serveur. La multiplication des schémas d'authentification au sein d'un protocole doit aussi prendre en compte les possibilités de négociation à la baisse (*downgrade*) et les prévenir.

2. Remote Procedure Call.

Échange de clé De nombreux protocoles sécurisés d'échange de données (SSH, TLS, etc.) supportent différents schémas d'échange de clé pour la mise en place de clés de session visant à protéger le trafic, en confidentialité et intégrité.

Certains de ces schémas d'échange basés sur des mécanismes asymétriques autorisent le déchiffrement ultérieur des sessions au possesseur de la clé privée du serveur. La compromission de cette clé remet donc dans ce cas en cause la confidentialité de l'ensemble des échanges précédemment réalisés. D'autres schémas, fournissant une propriété dite de PFS (*Perfect Forward Secrecy*), préviennent cette éventualité en décorrélant les clés de session du secret d'authentification. Cette propriété prévient le déchiffrement des sessions passées par un attaquant en possession du secret du serveur et nécessite la mise en œuvre d'attaques actives pour tirer un quelconque bénéfice de la possession de ce secret.

Confidentialité et intégrité des échanges Les protocoles d'administration à distance sont utilisés pour l'échange de tous types d'informations : des données pures mais également des informations de contexte et de contrôle sur le système voire le réseau administré, comme des mots de passe d'authentification à d'autres systèmes, des habitudes d'administration. La garantie forte de confidentialité de l'ensemble des informations échangées est donc une nécessité.

L'intégrité des échanges est également primordiale pour éviter la modification à la volée par un attaquant de commandes d'administration.

L'utilisation d'algorithmes de chiffrement et d'intégrité à l'état de l'art au sein de schémas éprouvés permet généralement de traiter cette problématique [6]. *A contrario*, le maintien pour des questions de rétrocompatibilité d'algorithmes dépassés ou l'utilisation de schémas propriétaires faibles peuvent avoir un impact fort sur la confidentialité ou l'intégrité des échanges.

Anti-rejeu Il est primordial de prévenir tout type de rejeu, aussi bien au niveau des mécanismes d'authentification comme évoqué précédemment qu'au niveau des échanges de trafic ou de signalisation. Le protocole doit également garantir le bon séquencement des messages à la réception, toujours dans le but d'éviter les modifications des échanges par un attaquant sur la base d'échanges précédents ou en cours.

Surface d'attaque Hormis les bonnes propriétés cryptographiques précédentes, le protocole doit également limiter la surface qu'il expose sur les

serveurs sur lesquels il est déployé. Ainsi, celui-ci devrait être développé en limitant les traitements réalisés, notamment avant authentification. Dans une démarche de défense en profondeur, les liens avec les différents éléments critiques du système devraient être limités par conception, pour éviter les impacts de vulnérabilités éventuelles.

Sécurité par défaut et simplicité de configuration La simplicité de configuration côtés client et serveur, ainsi que la mise en œuvre de paramètres solides par défaut est essentielle pour éviter les surprises.

De plus, les éléments de configuration accessibles à la fois côté client et côté serveur devraient permettre à l'administrateur de configurer facilement les différents aspects de sécurité du protocole, et notamment les paramètres associés à l'authentification, le chiffrement et l'intégrité. L'interface devrait également fournir un statut clair sur les mécanismes sélectionnés.

2 Présentation fonctionnelle de RDP

2.1 Présentation générale du protocole

Cet article n'a pas vocation à détailler le fonctionnement du protocole et se limite donc à une introduction : en effet, les bases de RDP sont spécifiées dans un document de plus de 400 pages [28] renvoyant lui-même vers plusieurs dizaines de spécifications Microsoft, UIT [3] et IETF [4]. Ce document décrit également les différents mécanismes de sécurité utilisés par les différentes évolutions de RDP. Au total, l'ensemble des spécifications publiées par Microsoft associées à RDP et à ses extensions dépasse les 2000 pages.

Le protocole RDP vise initialement le transport :
- des entrées utilisateur ;
- des données d'affichage du système distant au système local.

Additionnellement, un nombre important de fonctionnalités ont été ajoutées au protocole, permettant notamment le déport de divers périphériques du client vers la machine distante (port série, port parallèle, lecteur de SmartCard, port USB, entrées/sorties audio, etc.). Les échanges entre

3. *Union Internationale des Télécommunications.*
4. *Internet Engineering Task Force.*

presse-papiers local et distant ou encore l'impression côté client font également partie des améliorations protocolaires. La manière dont le protocole supporte celles-ci est détaillée par la suite.

Les entrées utilisateur se réduisent initialement à des évènements souris et clavier. Pour les transporter, RDP utilise deux types de PDU [5]. Le premier, dénommé *Slow-Path* est calqué sur des PDU du protocole UIT T.128. Le second, plus courant en pratique, est optimisé en taille. Dénommé *Fast-Path*, il a été introduit par Microsoft dans la version 5.0 de RDP pour optimiser la bande passante. Ces PDU transportent les frappes clavier (*keycodes*, enfoncement/relâche de touche) et les déplacements de souris (coordonnées de curseurs), ainsi que des signaux de synchronisation.

En retour, une fois les commandes du client relayées au niveau du serveur, celui-ci transmet des mises à jour graphiques à destination du client. Ces éléments, dénommés *Bitmap Updates*, sont également transportés dans des PDU *Slow-Path* et *Fast-Path* introduits précédemment. Une fois traités par le client, ils produisent des mises à jour locales des données d'affichage de la session utilisateur. Ces données graphiques bénéficient d'une compression spécifique RLE [6].

Outre les éléments graphiques, le protocole offre également un moyen de compresser les autres données échangées avec MPPC [7] [32]. D'un *buffer* de 8Ko dans la version initiale (RDP 4.0), celui-ci a été étendu pour passer à 64Ko dès RDP 5.0.

De plus, divers mécanismes de sécurité permettent en fonction du paramétrage de chiffrer et protéger en intégrité tout ou partie des données, dès le premier paquet échangé ou après une séquence de négociation. Ces mécanismes sont détaillés en section 4.

Largement basé sur une sous-partie des protocoles du standard UIT T.120, RDP reprend de ceux-ci un mécanisme extensible de canaux virtuels permettant les communications entre composants du système local et du système distant. Ce mécanisme est utilisé notamment pour le transport des entrées utilisateurs et des mises à jour graphiques. Il a permis de nombreuses évolutions du protocole au fil des versions, et notamment le support de nouvelles fonctionnalités.

2.2 Vue d'ensemble

La figure 1 donne une vue d'ensemble de RDP, présentant les trois facettes principales du protocoles :

5. *Protocol Data Unit.*
6. *Run Length Encoding, algorithme de compression graphique Microsoft.*
7. *Microsoft Point-to-Point Compression Protocol.*

1. les différentes fonctionnalités supportées par RDP, amenant chacune sa part de complexité à l'ensemble protocolaire, par des liens supplémentaires avec les différents couches du système ;

2. la "plomberie" protocolaire, permettant l'échange d'informations entre clients et serveurs. La concision obtenue par l'utilisation de protocoles UIT (relativement complexes et basés sur ASN.1) et de divers mécanismes de compression participe à la complexité de l'ensemble ;

3. les mécanismes de sécurité intégrés au protocole évoluent depuis plus de 15 ans. Les décisions de conception et de configuration par défaut associées aux besoins de rétrocompatibilité avec les systèmes déployés rendent leur compréhension et leur configuration difficiles.

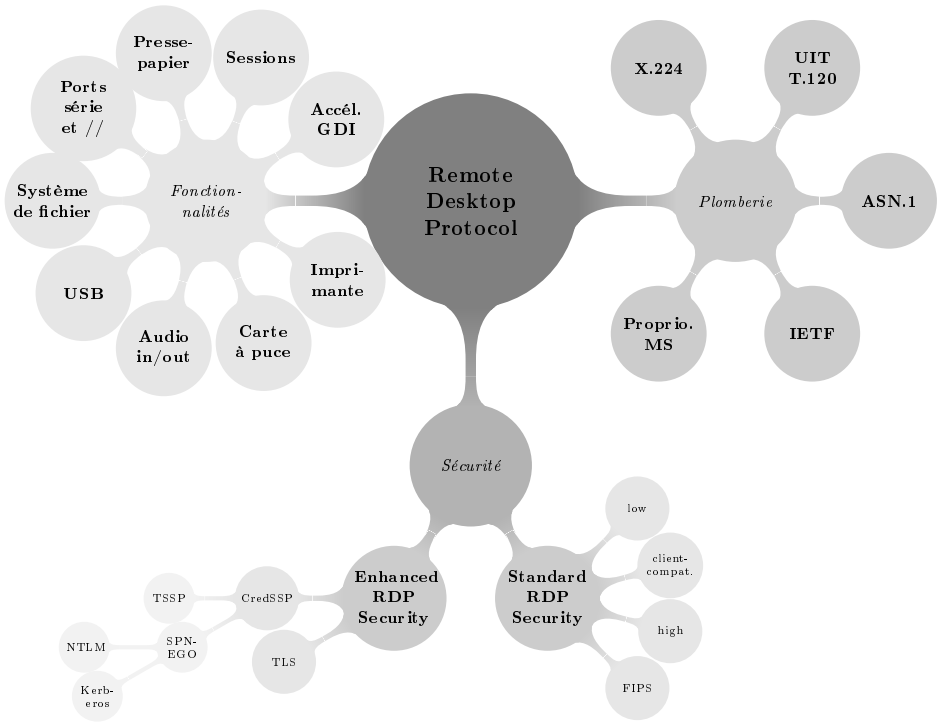

Figure 1. Vue d'ensemble du protocole

2.3 Détails protocolaire

Encapsulations protocolaires Cette sous-section étend la précédente en présentant (très succinctement) une séquence de connexion RDP. Elle intéressera principalement le lecteur ébahi devant le volume important de données échangées dans une trace réseau capturée [8].

Pour être utilisable sur les réseaux IP, RDP est transporté au dessus de TCP. Par défaut, le serveur est en écoute sur le port 3389.

Le transport de protocoles ISO au dessus de TCP passe par une couche d'abstraction dénommée TPKT, qui émule le service de transport ISO COTP. En pratique, celle-ci se matérialise par la présence au dessus de TCP d'un simple header de 4 octets renseignant (sur deux octets) la longueur du paquet transporté. Les paquets X.224 qui constituent la base du protocole sont donc ensuite transportés dans ces paquets TPKT.

Après un échange permettant la montée de la connexion X.224 (X.224 Connection Request PDU du client au serveur et X.224 Connection Confirm PDU du serveur au client), les paquets échangés par TPKT sont ensuite tous des paquets de données appelés X.224 Data PDU.

Cette connexion X.224 sert au transport du protocole T.125 [9]. Un des aspects fondamentaux de ce protocole concerne sa capacité, après une phase initiale de connexion, à transporter différents canaux d'échanges de données, dénommés canaux virtuels [10] (*virtual channel*). Cette fonctionnalité fournie par les bases du protocoles est celle qui a permis de supporter les nombreuses évolutions de RDP.

Ces canaux transportent notamment les entrées claviers et les déplacement de souris depuis la première version du protocole. Le déport du presse-papier ou le déport d'impression ont ainsi chacun été associés à un canal virtuel spécifique.

D'un point de vue protocolaire, les paquets échangés prennent donc la forme générique suivante :

```
Client                                                    Serveur
                              ...
   ---- IP/TCP/TPKT/X.224Data/T.125-SendDataRequest ----->
   <--- IP/TCP/TPKT/X.224Data/T.125-SendDataIndication ---
   ---- IP/TCP/TPKT/X.224Data/T.125-SendDataRequest ----->
```

8. Contrairement à Wireshark, les versions récentes de Network Monitor sont capables d'aller assez loin dans la dissection du protocole.

9. *Multipoint Communication Services Protocol* (MCS), de la suite T.120 de l'UIT.

10. L'article ne fait volontairement pas la distinction entres canaux virtuels statiques et dynamiques.

```
<--- IP/TCP/TPKT/X.224Data/T.125-SendDataIndication ---
                              ...
```

Dans la terminologie RDP, les paquets décrits précédemment sont dénommés *Slow Path*.

Pour diminuer la bande passante consommée et améliorer la réactivité, Microsoft a introduit à partir de RDP 5.0 (Windows 2000) des optimisations permettant de supprimer une partie de l'encapsulation pour certains paquets courts comme les entrées souris/claviers et les mises à jour graphiques.

Ces optimisations prennent la forme d'un nouveau type de paquet, dénommé *Fast-Path*, qui suppriment les headers TPKT, X.224 et T.125 pour transporter de manière compressée leur contenu.

Exemple de montée de session La liste ci-dessous présente séquentiellement les 8 étapes[11] et type de messages échangés lors d'une montée de session RDP. Elle reprend ainsi la séquence décrite en section 1.3.1 de [28], en se concentrant sur les aspects sécurité.

1. **Connection Initiation**

   ```
   Client                                                  Serveur
      ---- X.224 Connection Request PDU ----------------->
      <--- X.224 Connection Confirm PDU ------------------
   ```

   L'échange de ces paquets X.224 permettent d'initier la connexion. Tous les paquets échangés par la suite seront encapsulés dans des paquets de données X.224.

   Le premier paquet contient la liste des protocoles de sécurité supportés par le client, sous la forme d'un champ de bits, autorisant le choix des flags suivants :
   - `PROTOCOL_RDP` : *Standard RDP Security*
   - `PROTOCOL_SSL` : *TLS 1.0*
   - `PROTOCOL_HYBRID` : *CredSSP*[12]

   La réponse du serveur contient sa décision sur le mécanisme sélectionné (*Standard* ou *Enhanced RDP Security*). Dans ce cadre :
   - si *Enhanced RDP Security* a été sélectionné (*CredSSP* ou *TLS*), une session TLS est montée entre le client et le serveur, et les messages suivants sont donc échangés sous la protection de cette session ;

11. Les noms anglais des étapes ont été conservés pour plus de lisibilité.
12. Dans ce cas, le flag `PROTOCOL_SSL` est aussi positionné (CredSSP utilise TLS).

– si *Standard RDP Security* a été sélectionné, les échanges continuent en clair jusqu'à la fin de la négociation. Celle-ci est décrite ci-dessous.

2. **Basic Settings Exchange**

```
Client                                                    Serveur
   ---- MCS Connect Initial PDU with ----------------->
        GCC Conference Create Request
   <--- MCS Connect Response PDU with -----------------
        GCC Conference Create Response
```

Ces deux paquets T.125 sont encodés en ASN.1. Les données utilisateurs transportées dans le premier paquet (émis par le client) contiennent notamment des informations sur la machine cliente (taille de l'écran, type de clavier, *layout*, etc). Le protocole de sécurité sélectionné précédemment par le serveur y est également répété (*Standard RDP Security* ou *Enhanced RDP Security*).

Par ailleurs, les données utilisateurs transportées par le paquet client incluent également, *via* le champ **encryptionMethod**, les combinaisons d'algorithmes et de tailles de clés supportées par le client pour le mode *Standard RDP Security* :

– clés de 40 bits pour RC4 et le MAC ;
– clés de 56 bits pour RC4 et le MAC ;
– clés de 128 bits pour RC4 et le MAC ;
– triple DES avec clés de 168 bits et HMAC-SHA1 (Mode FIPS).

Le choix de la méthode de protection (algorithme et taille de clés) sélectionnée par le serveur dans le second paquet dépend de sa configuration (niveau *low*, *client-compatible*, *high* ou *FIPS*). Ce paquet intègre également pour le mode *Standard RDP Security* le *nonce* serveur (32 octets, émis en clair) à utiliser pour la dérivation de clé, ainsi que la clé publique "signée" [13] du serveur. Cette clé sera utilisée par le client à l'étape 4 pour la transmission chiffrée de son *nonce* (32 octets), participant également à la dérivation de clé.

3. **Channel Connection**

```
Client                                                    Serveur
   ---- MCS Erect Domain Request PDU ----------------->
   ---- MCS Attach User Request PDU ------------------>
   <--- MCS Attach User Confirm PDU -------------------
   ---- MCS Channel Join Request PDU ----------------->
```

13. Voir la description du **CVE-2005-1794.** en section 3.8.

```
<--- MCS Channel Join Confirm PDU ------------------
---- MCS Channel Join Request PDU ----------------->
<--- MCS Channel Join Confirm PDU ------------------
                        ...
---- MCS Channel Join Request PDU ----------------->
<--- MCS Channel Join Confirm PDU ------------------
```

Les 3 premiers paquets sont associés au protocole T.125 et n'ont pas d'intérêt particulier pour la discussion. Chaque échange de `MCS Channel Join Request PDU` et `MCS Channel Join Confirm PDU` permet au client RDP de l'utilisateur de joindre les canaux virtuels de son choix.

Suite à cette étape, l'ensemble des messages échangées entre le client et le serveur sont respectivement des **MCS Send Data Request** et **MCS Send Data Indication**.

Pour le moment, si *Standard RDP Security* a été choisi, les PDU échangées ne sont toujours pas protégées.

4. **RDP Security Commencement**

```
Client                                                Serveur
   ---- Security Exchange PDU ------------------------>
```

Ce paquet transmis par le client contient son *nonce* (32 octets) chiffré avec la clé publique transmise précédemment par le serveur. Après ce paquet, les échanges sont protégés [14] avec les clés dérivées des *nonces*. La sécurité de la session dépend donc de l'impossibilité pour un attaquant de remonter au *nonce* transmis chiffré par le client. Cette sécurité est donc directement liée à la taille de clé RSA utilisée (512 bits jusqu'à Windows Server 2003, 2048 bits pour les version plus récentes), et indirectement à l'authentification.

5. **Secure Settings Exchange**

```
Client                                                Serveur
   ---- Client Info PDU ------------------------------>
```

Ce paquet est donc le premier paquet chiffré. Le client y passe des informations comme le nom de l'utilisateur, le domaine. Une option du client RDP permet de retenir le mot de passe [15] ; dans ce cas, celui-ci transite dans ce paquet pour l'authentification de l'utilisateur sur le système distant, pour permettre l'*autologon*.

14. Avec le niveau *low*, le sens serveur vers client n'est pas protégé.
15. Ou le code PIN dans le cas d'utilisation de carte à puce.

6. **Licensing**

```
Client                                                    Serveur
   <--- License Error PDU - Valid Client --------------
```

Ce message est utilisé pour le transfert d'une licence du serveur au client, dans le cas où RDP est utilisé avec un serveur de licence.

7. **Capabilities Exchange**

```
Client                                                    Serveur
   <--- Demand Active PDU -----------------------------
   ---- Confirm Active PDU --------------------------->
```

Cet échange permet au client et au serveur de se synchroniser sur les fonctionnalités supportées (taille de buffer de compression MPPC, niveau de support GDI+, etc).

Après cet échange, le client peut commencer à émettre vers le serveur des entrées souris et clavier.

8. **Connection Finalization**

```
Client                                                    Serveur
   ---- Synchronize PDU ------------------------------->
   ---- Control PDU - Cooperate ----------------------->
   ---- Control PDU - Request Control ----------------->
   ---- Persistent Key List PDU(s) -------------------->
   ---- Font List PDU --------------------------------->
   <--- Synchronize PDU -------------------------------
   <--- Control PDU - Cooperate -----------------------
   <--- Control PDU - Granted Control -----------------
   <--- Font Map PDU ----------------------------------
```

Ces échanges permettent notamment la synchronisation graphique. Après réception du paquet `Font List PDU`, le serveur peut commencer l'émission de données graphiques au client.

3 Historique de RDP

Le protocole RDP est apparu avec Windows NT Server 4.0, dans sa version *Terminal Server Edition* (TSE), avec l'objectif initial de supporter le déport d'affichage dans des solutions de clients légers.

La fonctionnalité de "bureau à distance" permettant d'administrer graphiquement un système Windows s'est ensuite démocratisée avec Windows 2000. Depuis, le protocole a évolué régulièrement, au fil des différentes versions et *service pack* (SP) de Windows : support du déport

d'imprimantes (2000), du déport du son (XP), introduction de TLS (2003 SP1), *Network Level Authentication* (NLA, depuis Vista), etc.

Sans prétendre à l'exhaustivité, la liste ci-dessous, basée sur de nombreuses sources [8,9,7,1], retrace l'historique des versions du protocole en donnant notamment les versions de systèmes avec lesquels ils ont été introduits, ainsi que les améliorations fonctionnelles majeures et les évolutions sur le plan de la sécurité.

Il est important de noter que si la version de RDP qu'un serveur supporte est intimement liée à la version de système sous-jacent, il est possible sur un système client de mettre à jour son client RDP. Ainsi, Windows XP supporte par exemple des clients RDP 6.0 et RDP 7.0. En revanche, le serveur RDP disponible sur un Windows Server 2003 (version 5.x) ne pourra pas évoluer vers une version 6.x.

3.1 RDP 4.0 (Windows NT Server 4.0 TSE)

Introduite avec Windows NT Server 4.0 TSE [3], en 1996, il s'agit de la première version du protocole RDP. A l'époque, cette version reste assez limitée fonctionnellement : elle a pour principal objectif de fournir une solution de déport d'affichage sur un terminal utilisateur et de permettre le transport chiffré des entrées souris et clavier de cet utilisateur.

Le protocole offre le choix de trois niveaux de chiffrement – *low*, *medium* (défaut), *high* – décrits en section 4. Avec RDP 4.0, ces différents niveaux reposent tous sur l'utilisation d'un chiffrement RC4 40 bits. L'intégrité est assurée *via* un pseudo-HMAC [16].

Les mécanismes d'authentification et d'échange de clé intégrés au protocole (*Standard RDP Security* décrit en section 4.1) sont à cette époque inefficaces et vulnérables à des attaques par le milieu. Ils autorisent également le déchiffrement du trafic échangé au cours d'une session. Ces faiblesses ne seront corrigées que près de 10 ans plus tard, à partir de la version 5.2, introduite avec Windows Server 2003 SP1 (*Enhanced RDP Security* décrit en section 4.2).

3.2 RDP 5.0 (Windows 2000)

Introduite avec Windows 2000, cette version apporte de nouvelles fonctionnalités : cache local persistant de *bitmap*, reprise de session, impression locale, partage de presse-papier, optimisations réseau.

16. Ce pseudo-HMAC est décrit en section 5.3.6.1 de [28]

A cette époque, Windows 2000 supporte une taille de clé de 56 bits pour le chiffrement RC4.

L'installation du *Windows 2000 High Encryption Pack*[17] [10] ou du SP2 permet à l'époque d'activer ensuite l'utilisation d'une taille de clé de 128 bits.

En pratique, il faudra de toute manière attendre au moins le SP2 pour que les données des canaux virtuels – transportant notamment les mises à jour du presse-papiers et les impressions – soient également protégées en confidentialité et intégrité [13].

Par ailleurs, jusqu'à correction par le SP4 de la vulnérabilité MS02-051 [18] (détaillée en section 3.8), il est important de noter que les touches claviers sont trivialement accessibles à un attaquant.

3.3 RDP 5.1 (Windows XP)

Introduite avec Windows XP Professionnel, en 2001, cette version apporte notamment le support des couleurs sur 24 bits, la redirection de disque, de l'audio et des ports COM. Elle apporte également le support de l'ouverture de session par carte à puce.

Un client 5.1 est disponible pour Windows 2000, Windows 9x, Windows NT 4.0. C'est avec cette version que le nom du client est modifiée de *Terminal Services Client* pour devenir *Remote Desktop Client*.

Du point de vue de la sécurité, cette version ajoute aux trois niveaux basés sur RC4 introduits précédemment un nouveau niveau, *FIPS*, utilisant Triple DES pour le chiffrement et HMAC-SHA1 pour l'intégrité.

Le bug MS02-051 [18] rendant trivial l'accès à un attaquant aux touches clavier échangés durant la session est corrigé à partir du SP1, sorti fin 2002.

3.4 RDP 5.2 (Windows Server 2003 SP1)

Cette version introduite avec Windows Server 2003 SP1[18] permet l'utilisation de TLS pour la protection des sessions RDP : l'un des bénéfices de cet ajout est la capacité pour le client d'authentifier le serveur. De plus, l'utilisation de TLS pour la protection des flux RDP permet d'envisager l'utilisation de modes d'échange de clés, de chiffrement et d'intégrité éprouvés.

17. À installer avant le client : http ://support.microsoft.com/kb/257894/en-us
18. Un Server 2003 sans SP1, également en version 5.2, ne permet pas de configurer TLS.

3.5 RDP 6.0 (Windows Vista)

Cette version a été introduite avec Windows Vista, en 2007. Des versions de clients RDP 6.0 sont disponibles pour Windows Server 2003 et Windows XP SP2.

D'un point de vue fonctionnel, cette version ajoute notamment le support d'un mode couleur 32 bits et le support du multi-écrans.

Une des évolutions majeures en matière de sécurité apportée par RDP 6.0 concerne l'intégration de NLA, détaillé en section 4.2 [19]. Fonctionnellement, NLA permet de fournir directement les identifiants utilisateurs au serveur, permettant ainsi de réaliser l'authentification au tout début de la session RDP. Visuellement, la mire d'authentification n'apparaît donc plus lorsque NLA est utilisé.

Sous Windows XP, il faut noter que le support client de CredSSP n'est fonctionnel qu'après passage au SP3, activation [20] du mécanisme CredSSP et après installation de la version 6.1 ou supérieure du client RDP [21].

3.6 RDP 6.1 (Windows Server 2008)

Cette version a été introduite avec Windows Vista SP1 et Windows Server 2008. Des mises à jour clientes pour cette version sont disponibles pour Windows Vista SP1 et Windows XP SP2.

3.7 RDP 7.0 (Windows 7)

Cette version a été introduite avec Windows 7 et Windows Server 2008 R2. Des mises à jour sont disponibles pour les clients pour Windows XP SP3, Windows Vista SP1 et SP2. D'un point de vue fonctionnel, cette version apporte notamment le support audio bidirectionnel, une accélération des échanges de *bitmap*, le support d'Aero et le support d'un vrai mode multi-écrans.

En matière de sécurité, un certificat auto-signé RSA de 2048 bits d'une validité de 6 mois est généré par le système et utilisé par défaut pour la session TLS. Les détails sont discutés en section 4.2.

3.8 Panorama historique des vulnérabilités

Depuis la première version de RDP, les implémentations Microsoft (client, serveur) ont connu près d'une quinzaine de vulnérabilités référencées. Celles-ci sont reprises ici et classées en trois catégories principales.

19. En pratique, l'implémentation de NLA est réalisée par CredSSP.
20. `http://support.microsoft.com/kb/951608`
21. `http://support.microsoft.com/kb/951616`

Les bulletins de sécurité Microsoft sont utilisés de préférence comme référence mais sont remplacés par les CVE, KB et autres annonces associées aux vulnérabilités découvertes lorsque Microsoft n'a pas produit de bulletin de sécurité.

Déni de service
- MS01-006 [14], MS01-040 [15], MS01-052 [16], MS05-041 [22], MS11-065 [27] ;

Exécution de Code
- MS02-046 [17], MS09-044 [23], MS11-017 [25], MS11-061 [26], MS12-020 [30] ;

Conception/Crypto
- **MS02-051** [18] : *Cryptographic Flaw in RDP Protocol can Lead to Information Disclosure*
- **KB275727** [13] : *High Encryption on a Remote Desktop or Terminal Services Session Does Not Encrypt All Information*
- **Forsberg03** [20] : *Microsoft Terminal Services vulnerable to MITM-attacks*
- **CVE-2005-1794** [21] : *shared RSA key is embedded in RDP library mstlapi.dll*

Les quelques dénis de services référencés sont probablement le reflet de la complexité du protocole et de la difficulté à en réaliser une implantation sûre. Les liens étroits avec les nombreux éléments critiques du système apparaissent également au travers des vulnérabilités autorisant une exécution de code. Mais la partie la plus intéressante de cet historique concerne les quatre vulnérabilités placées ci-dessus dans la catégorie Conception/Crypto. Celles-ci sont détaillées ci-après :

- **MS02-051** [18] : le bulletin de sécurité, publié en septembre 2002 [22], référence deux vulnérabilités affectant Windows 2000 (RDP 5.0) et Windows XP (RDP 5.1).
  L'une d'elle est une **erreur de conception cryptographique générique** décrite par Hugo Krawczyk dans [31], connue sous le nom *authenticate-and-encrypt*. Elle consiste à calculer un motif d'intégrité sur le texte clair puis à chiffrer le clair et à transmettre les deux résultats. Deux clairs identiques possèdent alors le même motif d'intégrité. Dans le cas de RDP, il s'agit du schéma utilisé depuis la conception du protocole.

22. Microsoft a été notifié de la vulnérabilité le 16 avril 2002.

A partir de RDP 5.0, les entrées clavier de l'utilisateur, sont transportées dans des PDU spécifiques (*Fast Path PDU*) de taille réduite : 2 octets d'en-tête, 8 octets de MAC et 2 octets de données chiffrées. Les 2 octets chiffrés contiennent le type d'évènement clavier (touche enfoncée, touche relâchée) et la touche concernée.
Si suffisamment de trafic a été échangé, un attaquant est donc capable, par une analyse fréquentielle des valeurs de MAC, d'en déduire les touches enfoncées, et donc le texte clair.

- **KB275727** [13] : Par défaut, **avant le SP2 pour Windows 2000**, les informations échangées par les canaux virtuels RDP (utilisés par le protocole pour certaines fonctionnalités) ne sont pas chiffrés par défaut. Cet oubli expose donc notamment les échanges de presse-papiers mais également les redirections d'impression.
- **Forsberg03** [20] : Début 2003, Eric Forsberg signale à Microsoft que le schéma utilisé pour l'échange de clé est trivialement sujet à une attaque par le milieu. Le serveur transmet en effet au client une clé publique RSA que celui-ci n'authentifie pas.
  La vulnérabilité est "corrigée" **silencieusement** par Microsoft en intégrant à ses implémentations une clé RSA fixe. La partie privée est utilisée côté serveur pour signer une seconde clé publique générée localement. La partie publique est utilisée côté client pour vérifier la signature sur cette seconde clé. Ce schéma nous amène donc à la vulnérabilité suivante.
- **CVE-2005-1794** [21] : La "correction" proposée pour la vulnérabilité ci-dessus, consistant à masquer une clé de signature/vérification dans une bibliothèque du système est évidemment découverte mais ne fait pas l'objet d'un bulletin de sécurité : en effet, elle n'a jamais été corrigée.
  Cette clé est aujourd'hui référencée dans la spécification de RDP [28] (en section 5.3.3.1.1) et constitue toujours aujourd'hui le seul "mécanisme d'authentification" du mode *Standard RDP Security* (détaillé en section 4.1) pour les serveurs RDP déployés sur les systèmes jusqu'à Windows XP inclus.
  Même si des alternatives sont disponibles sur les systèmes Microsoft plus récents, des arguments de rétrocompatibilité font que ce mécanisme reste utilisable sur certains d'entre eux.

4 Mécanismes de sécurité de RDP

RDP offre deux modes principaux de sécurité pour la protection des échanges.

Le premier, dénommé *Standard RDP Security*[23], correspond au mode de protection historique intégré au protocole. Contrairement au second mode, qui réutilise en grande partie des schémas connus, celui-ci repose sur l'utilisation directe de primitives cryptographiques. Il offre quatre "niveaux" de protection différents, que les clients et serveurs peuvent négocier.

Le second mode, dénommé *Enhanced RDP Security*[24], correspond à l'utilisation pour la protection des échanges RDP d'un protocole externe parmi les deux disponibles, TLS (à partir de 2003 SP1) ou CredSSP (à partir de Vista). Cette approche permet à RDP de reposer sur des protocoles éprouvés pour la sécurité des échanges.

4.1 *Standard RDP Security*

Introduction Comme évoqué précédemment, *Standard RDP Security* supporte quatre niveaux de protection en confidentialité des échanges entre le client et le serveur.

Tous les niveaux de chiffrement hormis FIPS utilisent l'algorithme RC4, seule la taille de clé peut changer entre 40, 56 ou 128 bits. La taille de clé négociée est également utilisée pour la clé d'authentification des paquets (MAC). Le niveau FIPS utilise les algorithmes *Triple DES* et *HMAC-SHA1*.

Low Le premier niveau, dénommé *low*, n'impose que le chiffrement des données du client vers le serveur, le canal inverse restant en clair. L'objectif est ici de protéger les données d'authentification émises par le client et notamment la saisie de son mot de passe dans la GINA[25] ou *via* les *Credentials Providers* depuis RDP 6.0 (Vista). La taille de clé selectionnée par le serveur est la plus grande supportée par le client.

> En niveau *low*, les données provenant du serveur sont considérées comme "moins confidentielles" et ne sont pas chiffrées.

23. *Standard RDP Security* est spécifié en section 5.3 de [28]
24. textitEnhanced RDP Security est spécifié en section 5.4 de [28]
25. *Graphical Identification and Authentication*, la boîte de dialogue d'authentification avant Vista

Client compatible Le second niveau, "*client compatible*", étend la protection aux deux sens de communications, toujours avec la taille de clé la plus grande supportée par le client.

High Le troisième niveau, "*high*", impose également un chiffrement des deux sens de communication, mais cette fois-ci en utilisant la taille de clé la plus grande supportée par le serveur. Les clients qui ne supportent pas les algorithmes proposés par le serveur ne sont pas autorisés à se connecter.

FIPS Le dernier niveau est une extension du troisième niveau imposant l'utilisation de primitives de chiffrement validées FIPS 140-1 [12]. C'est à dire *Triple DES* pour le chiffrement et *HMAC-SHA1* pour le MAC.

En pratique, un cinquième niveau de chiffrement existe, qui correspond à une absence de chiffrement des flux.

Le tableau suivant résume les différents niveaux :

Niveau	C⇒S	S⇒C	*méthode*	*algo*	*MAC*
Low	chiffré	**clair**	**max client**	RC4	MD5/SHA1
Client Compatible	chiffré	chiffré	**max client**	RC4	MD5/SHA1
High	chiffré	chiffré	max serveur	RC4	MD5/SHA1
FIPS	chiffré	chiffré	n/a	3DES	HMAC-SHA1

Figure 2. Résumé des principaux paramètres de sécurité associés au choix du niveau de chiffrement RDP (*Encryption Level*)

Authentification Il convient de distinguer les deux modes de fonctionnement : le mode autonome et le mode *Terminal Server*. Le mode autonome est le mode par défaut permettant la montée de deux sessions concurrentes, et ne nécessitant pas la mise en œuvre de serveur de licence. Il s'agit de celui considéré dans l'article.

En mode autonome, le mécanisme d'authentification proposé de base par le protocole est en pratique inexistant :

- le client n'est pas authentifié par le serveur ;
- le serveur présente sa clé publique, signée par une clé privée connue de tous [20,21] car spécifiée dans la documentation Microsoft [28].

Échange de clés Par ailleurs, le mécanisme d'échange de clé utilisé repose sur un simple chiffrement par la clé publique RSA du serveur d'un *nonce* de 32 octets produit par le client (appelé *ClientRandom*). Le serveur envoie également un *ServerRandom* de 32 octets (non chiffré) lors de l'échange.

Ces valeurs sont ensuite utilisées pour dériver trois clés de session de 128 bits chacune, à l'aide des fonctions de hachage MD5 et SHA-1.

- la clé de chiffrement client vers serveur ;
- la clé de chiffrement serveur vers client ;
- la clé d'authentification utilisée pour le MAC.

Dans le cas où la taille de clé négociée est inférieure à 128 bits, seule une partie de ces 128 bits sont utilisés, le reste étant remplacé par des valeurs fixes.

Si un attaquant peut décrypter le *ClientRandom*, alors celui-ci pourra accéder à l'intégralité des communications. La sécurité de la transaction dépend donc de la taille de la clé publique RSA utilisée ; la taille minimale recommandée par le Référentiel Général de Sécurité (RGS) [6] est 2048 bits.

Intégrité Lors de la négociation à l'établissement de la connexion, à partir de RDP 5.2, le client et le serveur peuvent négocier l'utilisation de *salted MAC* au lieu d'un simple MAC. Cette option intègre un compteur de chiffrement / déchiffrement dans le calcul du MAC des paquets.

En mode normal, le MAC du paquet est calculé de la manière suivante :

```
Pad1 = 0x36 répété 40 fois pour obtenir 320 bits
Pad2 = 0x5C répété 48 fois pour obtenir 384 bits
SHAComponent = SHA(MACKeyN + Pad1 + DataLen + Data)
MACSignature = First64Bits(MD5(MACKeyN + Pad2 + SHAComponent))
```

`MACKeyN` est soit `MACKey40`, `MACKey56` ou `MACKey128`, selon la taille de clé négociée. Le mode *salted MAC* intègre un compteur supplémentaire, `EncCount` :

```
SHAComponent = SHA(MACKeyN + Pad1 + DataLen + Data + EncCount)
MACSignature = First64Bits(MD5(MACKeyN + Pad2 + SHAComponent))
```

Ici `EncCount` représente le nombre d'opérations de chiffrement qui ont été réalisées jusqu'ici, dans le sens de communication concerné (C⇒S ou S⇒C).

Analyse de sécurité L'analyse ne concernera que les niveaux de chiffrement *compatible client* et *niveau élevé*, qui sont ceux généralement rencontrés. Les niveaux faible et FIPS ne sont en pratique jamais rencontrés car il faut les configurer explicitement.

Authentification Comme précisé précédemment, l'authentification du serveur est inexistante. En effet, la clé publique du serveur est signée par une clé privée connue de tous.

Échange de clé Si le mécanisme de dérivation de clé à partir des données aléatoires échangées ne présente pas de vulnérabilité évidente, il n'en va pas de même pour l'échange. En effet, le mécanisme de transport de clé ne possède pas la propriété de *Perfect Forward Secrecy* et donc la compromission de la clé privée du serveur permet la compromission de toutes les communications observées, même auparavant.

Ainsi, l'utilisation de clés publiques RSA de 512 bits sur les serveurs RDP anciens (jusqu'à XP et 2003) ne permet absolument pas de garantir la confidentialité des échanges. En effet, les clés de cette taille peuvent être factorisées très rapidement, y compris avec peu de moyens de calcul [6,19].

Un attaquant ne pouvant pas réaliser d'attaque active (MITM) mais disposant de capacités d'écoute est donc capable de décrypter les échanges et la complexité protocolaire reste donc le seul rempart empêchant l'accès pratique aux informations échangées : frappes clavier (y compris les mots de passe saisis), presse-papier, données graphiques et autres canaux de communication.

Chiffrement L'utilisation de clés de 40 ou 56 bits, avec RC4, ne permet évidemment pas de garantir la sécurité, l'entropie de la clé étant très faible. L'idéal est donc d'utiliser des clés de 128 bits.

Intégrité D'une manière générale, on pourra noter l'utilisation d'un schéma spécifique et non d'un réel HMAC. De plus, la valeur résultante est tronquée à 64 bits. *A priori*, aucune attaque ne semble néanmoins réalisable en pratique.

Le mode normal ne dispose pas de protection anti-rejeu en tant que tel. Il devrait donc être en théorie possible de rejouer des paquets. En pratique, sans être une garantie de sécurité forte, l'utilisation d'un chiffrement par flot rend l'exercice difficile.

Néanmoins, le principal problème de ce mode de calcul d'intégrité n'est justement pas lié à la protection de l'intégrité mais à la confidentialité. En

effet, contrairement aux bonnes pratiques, le calcul d'intégrité se fait sur les données en clair et le MAC résultant n'est pas chiffré. Donc au cours d'une session, deux paquets de données identiques auront le même MAC. Ce qui est particulièrement problématique pour les frappes clavier : les *Fastpath PDU* sont très petites et il est tout à fait possible de retrouver les touches à partir du flux RDP sans attaque active.

Ce problème a été corrigé avec MS02-051 [18] avec l'introduction du mode *salted MAC*, qui en rajoutant un différenciateur, correspondant aux nombre de chiffrements effectués, permet d'obtenir un MAC différent pour deux clairs identiques, corrigeant ainsi la vulnérabilité du mode classique.

4.2 *Enhanced RDP Security*

Introduction Contrairement au mode historique et propre à RDP, *Standard RDP Security*, le mode *Enhanced RDP Security* délègue la protection en confidentialité et intégrité des échanges à un mécanisme externe. Le protocole RDP, en clair, sera donc encapsulé dans ce mécanisme.

Les mécanismes de protection externes sont les SSP (*Security Support Provider*). Sous Windows, ces bibliothèques permettent de réaliser une authentification, la dérivation de clés, puis la protection des messages. RDP supporte deux SSP :

- **Secure Channel** (ou Schannel) qui met en œuvre TLS pour l'authentification et la protection des échanges ;
- **CredSSP** qui met également en œuvre TLS pour transporter une authentification basée sur SPNEGO [29] [26] et la délégation des informations d'identification. Dans la pratique, CredSSP fait appel à d'autres SSP : Negotiate pour la négociation SPNEGO, (qui lui-même fait appel à NTLM ou Kerberos) et TSSSP [27] pour la délégation. La figure 3 présente l'imbrication de ces SSP.

L'utilisation d'un des deux mécanismes externes de sécurité peut être négociée (*Negotiation-Based Approach*) ou directe (*Direct Approach*).

La première approche a l'avantage d'être rétrocompatible avec le mécanisme historique ; la communication débute ainsi par l'échange, en clair, d'une requête et d'une confirmation de connexion X.224. Le mécanisme de sécurité externe est ensuite mis en oeuvre pour protéger le reste de la session.

La seconde approche permet la mise en oeuvre directe du mécanisme de sécurité externe, avant tout échange RDP. Elle favorise la sécurité à l'interopérabilité.

26. *Simple and Protected GSS-API Negotiation Mechanism Protocol Extensions.*
27. *TS Service Security Package.*

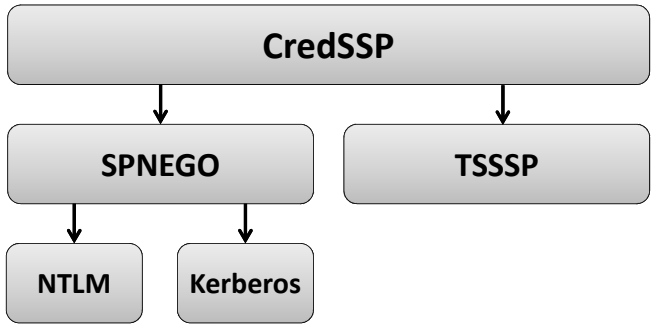

Figure 3. Imbrication des SSP dans CredSSP

TLS :

Lorsque TLS est utilisé, une session TLS est établie par le SSP entre le client et le serveur. Le tunnel TLS ainsi établi est ensuite utilisé pour la sécurisation en confidentialité et intégrité de l'intégralité des échanges de la session RDP.

L'apport majeur de l'utilisation de TLS est la possibilité de pouvoir authentifier le serveur *via* un certificat X.509. Celui-ci peut être auto-généré (ce qui est le cas par défaut depuis Windows Vista ou 2008) ou importé dans le magasin de certificats du serveur.

Si TLS propose l'authentification du serveur *via* un certificat, la validation de celui-ci est entièrement du ressort du client et dépendra de son paramétrage (stratégie et ancres de confiance configurées). Cette problématique est abordé en section 5.2.

CredSSP :

Credential Security Support Provider [24] est le SSP apparu avec Windows Vista/2008 [28]. Dans le principe, ce SSP implémentant un mécanisme d'authentification appelé NLA (*Network Level Authentication*) vise à renforcer l'authentification entre un client et un serveur puis à permettre la délégation des informations d'identification. Dans les faits, le service Terminal Server est le seul utilisateur actuel de CredSSP.

L'authentification et la délégation des informations d'identification *via* CredSSP sont réalisées en quatre phases présentées en figure 4 puis détaillées ci-dessous.

28. CredSSP est également disponible sous Windows XP Service Pack 3, mais n'est pas activé par défaut (cf. KB951608).

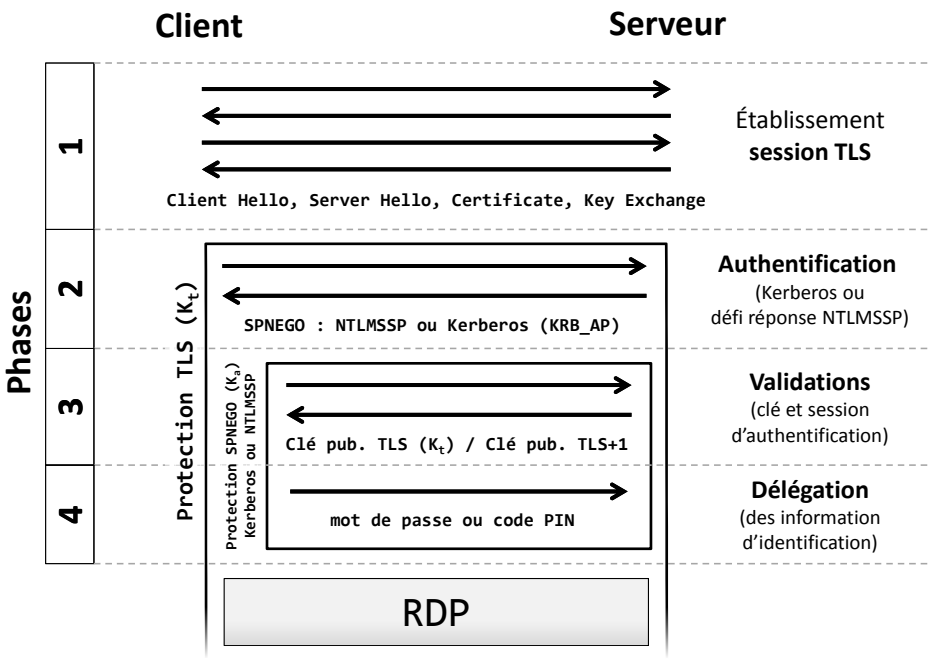

Figure 4. Les 4 phases de CredSSP

Phase 1 Dans la première phase, le client et le serveur établissent une session TLS. La mise en place de cette session permet, d'une part, au client d'authentifier **éventuellement** le serveur *via* le certificat X.509 qu'il présente et, d'autre part, de bénéficier d'un canal de communication protégeant tous les échanges suivants. La réalisation de cette session TLS est assurée par le SSP Secure Channel.

Phase 2 Après la mise en place de la session TLS, l'utilisateur doit s'authentifier auprès du serveur. Pour cela, CredSSP utilise le SSP Negotiate qui permet de négocier un protocole d'authentification : Kerberos ou NTLMSSP. Après authentification, le SSP correspondant au protocole choisi est en mesure de chiffrer des messages ce qui permet de mettre en place un second canal de chiffrement utilisé pour la suite des échanges[29].

Pour pouvoir utiliser ce canal de chiffrement, le serveur doit avoir accès d'une matière ou d'une autre au secret de l'utilisateur ou aux clés de chiffrement :

29. Le mécanisme de génération de clés et de chiffrement est différent suivant NTLMSSP ou Kerberos.

- si le compte utilisé par le client est un compte local, le serveur récupère dans sa base SAM locale l'empreinte NTLM du compte puis dérive les clés ;
- si le compte du client est un compte de domaine et que le SSP NTLM est utilisé, le serveur doit valider auprès d'un contrôleur de domaine le compte du client. Pour cela, le serveur établit avec un contrôleur de domaine un canal sécurisé (*Secure Channel*) pour valider la réponse du client puis le contrôleur de domaine lui transmet les clés de chiffrement. Pour établir le canal sécurisé, le serveur doit être membre du domaine (ou disposer au moins d'un compte machine) ;
- si le compte du client est un compte de domaine et que le SSP Kerberos est utilisé, le client doit obtenir auprès d'un KDC (un contrôleur de domaine) un ticket à présenter au serveur. Pour cela, à partir du nom référençant le serveur, le client demande un ticket de service au KDC où, dans sa base (*i.e.* l'Active Directory), un compte doit être associé au SPN [30] demandé. Pour le service Terminal Server, le SPN est de la forme `TERMSRV/nom serveur`. Le secret associé à ce compte (*i.e.* l'empreinte NTLM) est utilisé pour chiffrer le ticket de service transmis au client qui le transmet à son tour au serveur. Il est donc nécessaire que le serveur connaisse le secret utilisé par le KDC afin de déchiffrer le ticket et récupérer la clé de chiffrement. Dans le principe de l'authentification Kerberos, si le serveur peut utiliser la session chiffrée (ce qui sera fait à l'étape suivante), cela prouve qu'il connaît la même clé que celle connue par le KDC, donc que c'est le serveur légitime.

Si Kerberos, *via* CredSSP, est utilisé pour l'authentification, le serveur est authentifié. En revanche, l'utilisation de NTLMSSP ne permet pas d'authentifier le serveur.

À partir de cette étape, en plus d'être chiffrés par TLS, les échanges des étapes 3 et 4 sont chiffrés par NTLMSSP ou Kerberos, suivant la méthode d'authentification utilisée.

Phase 3 La troisième phase a deux objectifs. Le premier est de valider la clé publique contenue dans le certificat présenté par le serveur lors de la négociation TLS de la phase 1. Cette vérification permet de s'assurer que la clé n'a pas été substituée par une attaque de type *man in the middle*. Le second objectif est de s'assurer que les deux parties sont bien

30. Service Principal Name.

en possession de la même clé (K_a) liée à l'authentification de la phase précédente. Pour ce faire, le client commence par envoyer la clé publique K_t inclue dans le certificat X.509 reçu. Cet envoi est chiffré par la clé K_a (et TLS). Le serveur compare K_t à celle qu'il a envoyée, puis la renvoie incrémentée de 1, toujours chiffrée avec K_a (et TLS). Si les échanges ont abouti, les deux parties sont assurées que leur correspondant connaît la clé K_a et que la clé K_t n'a pas été altérée.

Phase 4 Enfin, la dernière phase consiste à déléguer les informations d'identification du client. Pour cela, CredSSP transfère au serveur les éléments d'authentification du client (son mot de passe ou son code PIN et la référence de carte à puce). Grâce à ces éléments, le serveur peut ouvrir une session interactive de l'utilisateur. Cette délégation peut être manuelle (le client doit saisir les éléments lors de la connexion au serveur) ou automatiquement (le mot de passe de l'utilisateur est transmis automatique au serveur)[31].

> Avec RDP, quelque soit le mode de sécurité utilisé, le mot de passe de l'utilisateur ou son PIN de carte à puce finit par être transmis au serveur.

La suite des échanges RDP reste protégée par la session TLS.

Analyse de sécurité

TLS Tout d'abord, sous Windows XP, le mode *Enhanced RDP Security* n'étant pas supporté, il n'est pas possible d'authentifier un système Windows auquel on accède en RDP.

À partir de Windows 2003 SP1, l'utilisation de TLS pour la protection des échanges RDP nécessite un minimum de configuration côté client et côté serveur, notamment pour paramétrer l'authentification et les mécanismes cryptographiques utilisés.

Pour activer la protection TLS avec RDP, un certificat X.509 doit être présent dans le magasin de certificats de l'ordinateur et sélectionné depuis l'interface de configuration du service Terminal Server.

À partir de Windows 2003 SP1, il est nécessaire d'importer un certificat X.509 issu d'une PKI[32] pour utiliser TLS. En général, cette opération

31. Cette fonctionnalité n'est pas activée par défaut.
32. Infrastructure de Gestion de Clé, *i.e.* IGC

est rarement effectuée par les administrateurs ; TLS n'est donc pratiquement jamais utilisé avec Windows 2003.

Depuis Windows Vista ou 2008, TLS est configuré pour être utilisé par défaut. Un certificat X.509 étant nécessaire, le système génère automatiquement un certificat autosigné au nom du serveur. Ce certificat étant autosigné, il n'est pas possible de le vérifier, donc d'authentifier le serveur. En revanche, sa présence permet de d'activer et d'utiliser TLS, **sans authentification**. L'authentification du serveur sera réalisée par Kerberos *via* CredSSP [33] dans le tunnel TLS, pour ensuite permettre la validation de la clé publique présentée initialement par le serveur dans son certificat. La session TLS sera ainsi en quelques sorte "post-authentifiée". Toute cette logique est mise en oeuvre pour supporter la philosophie Microsoft de limiter les besoins de configuration. En pratique, ceci impose tout de même l'utilisation de Kerberos et donc l'accès des clients au KDC.

Sur les éditions serveur de Windows, l'interface de configuration du service permet de choisir le certificat à utiliser : soit le certificat autosigné par défaut ou un autre certificat, par exemple, importé de sa propre PKI.

Afin d'authentifier correctement le serveur, le client RDP doit valider le certificat présenté. Les critères de validation sont les mêmes que pour n'importe quelle connexion TLS (nom du serveur, date de validité, chaîne de certification, rôles du certificat, etc.). L'implémentation du mécanisme de validation, qui détermine en particulier le comportement à adopter en cas d'échec de la validation, est fondamentale du point de vue de la sécurité. Cela peut être aucun avertissement, une simple alerte ou un refus de connexion en cas d'échec d'authentification. La partie 5.2 décrit l'implémentation du client Microsoft. Évidemment, si la validation du certificat a échoué, mais que le client continue la session TLS, rien n'assure, *via* le mécanisme des certificats X.509, que le serveur soit authentique.

Concernant l'authentification du client *via* un certificat X.509 – mécanisme disponible nativement dans TLS – celle-ci n'est tout simplement pas utilisée par Microsoft. Dans la conception de RDP, l'authentification du client n'est pas réalisée par TLS, mais par l'ouverture de session interactive sur le serveur ou par l'authentification SPNEGO si CredSSP est utilisé.

Concernant la négociation des algorithmes cryptographiques utilisés par TLS dans le cadre de RDP, aucune interface graphique ne permet de déterminer la liste des algorithmes utilisables. Cependant, il est possible de restreindre la liste des algorithmes et méthodes d'échange de clés proposés

33. Le lecteur attentif se sera rendu compte que NTLMSSP peut être choisi, auquel cas l'authentification échouera.

ou acceptés par le SSP Secure Channel, qui met en œuvre TLS. Cette configuration s'effectue *via* un ensemble de clé dans la base de registre [34].

Depuis Windows Vista, il est également possible, *via* les stratégies de groupe [35], de restreindre ou de changer l'ordre des suites de chiffrements proposées (côté client) ou acceptées (côté serveur).

Cependant, ces paramètres configurent le SSP Secure Channel de manière globale et impactent donc tous les services du système utilisant TLS au moyen de ce SSP. Dans la pratique, aucun administrateur ne se risquera à modifier ces paramètres de peur des conséquences fonctionnelles. En particulier, ces paramètres influent sur la bibliothèque `WinHTTP` utilisée par des composants tels qu'Internet Explorer, Windows Update, etc.

Par défaut, l'algorithme retenu (`TLS_RSA_WITH_AES_128_CBC_SHA` depuis Vista) pour la protection des échanges TLS ne permet pas d'obtenir la propriété de PFS. Au moyen du paramètre indiqué ci-dessus, il est possible de forcer un ordre précis pour les suites cryptographiques. Le principe consiste à configurer, sur le serveur, une liste d'algorithmes dont les premiers sont ceux basés sur une méthode de négociation des clés de type `ECDHE`. Ceci permet de privilégier le choix de ces protocoles par le serveur et d'obtenir ainsi la propriété de PFS. Cependant, les clients ne supportant `ECDHE` (comme Windows XP) pourront tout de même se connecter. La seule manière de pallier ce problème est de désactiver les suites non `ECDHE` : ce paramétrage affecte encore une fois l'ensemble des services du systèmes utilisant TLS.

CredSSP Comme vu en section 4.2, la première phase de CredSSP consiste en la mise en place d'une session TLS. Les problématiques de sécurité décrites ci-dessus s'appliquent donc également.

Cependant, l'utilisation de CredSSP apporte la réalisation d'une authentification SPNEGO (NTLM ou Kerberos), l'établissement d'un canal chiffré basé sur l'authentification réalisée, puis la validation de la clé publique contenue dans le certificat présenté par le serveur au client.

> Concernant l'authentification basée sur SPNEGO, il n'est pas possible de forcer l'utilisation de Kerberos par rapport à NTLMSSP, bien que ce dernier offre moins de protection.

Il n'est notamment pas possible au client, avec les protocoles LM ou NTLM, de valider l'identité du serveur auquel il se connecte. Ainsi, en cas

34. `HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL`
35. Configuration Ordinateur, Modèles d'administration, Réseau, Paramètres de configuration SSL

de compromission d'un compte de machine d'un domaine, un attaquant aura la possibilité, avec NTLMSSP, d'usurper l'identité de n'importe quel serveur.

En revanche, si Kerberos est utilisé, le client est assuré de l'identité du serveur auquel il se connecte. En effet, le contrôleur de domaine, qui joue le rôle de KDC, donne au client un ticket de service destiné au serveur indiqué par le client (le SPN recherché est de la forme `TERMSRV/nom serveur`). Ce ticket est ensuite présenté par le client au serveur dans la phase 2 des échanges CredSSP. Seul le serveur ayant la clé partagée avec le KDC peut déchiffrer les informations contenues dans le ticket et ainsi récupérer la clé de chiffrement du canal sécurisé utilisé dans la phase 3. Cela nécessite que le client et le serveur puissent réaliser leur authentification au moyen de Kerberos.

Cependant, l'utilisation de Kerberos impose que le client et le serveur soient membres d'un même domaine Active Directory. De plus, le client doit avoir un accès réseau avec le KDC, ce qui n'est pas toujours possible, par exemple dans le cadre d'un accès distant.

La phase 3 des échanges CredSSP (envoi clé publique, retour clé+1) a deux objectifs. D'une part, le serveur s'assure que le client a bien reçu la même clé publique que celle qu'il a envoyée dans le certificat X.509. D'autre part, elle permet au client et au serveur de s'assurer que leur homologue est bien en mesure d'utiliser le canal sécurisé SPNEGO. Si Kerberos a été utilisé pour la mise en place de ce canal, cela assure au client que le serveur a pu déchiffrer le ticket qu'il lui a envoyé et donc que le serveur est reconnu par le KDC.

4.3 Conclusion

Le mode *Enhanced RDP Security* apporte, outre l'utilisation de TLS pour la protection des échanges RDP, surtout la possibilité d'authentifier le serveur vis-à-vis du client. Cette authentification peut prendre deux formes :

- la première repose sur le certificat X.509 présenté par le serveur dans la mise en place de la session TLS ;
- la seconde repose sur l'utilisateur du protocole CredSSP, seulement si Kerberos est utilisé. Dans ce cas, le certificat X.509 n'est plus vérifié, mais seulement la clé publique contenue dans le certificat et le serveur est authentifié *via* Kerberos.

Si CredSSP utilise NTLMSSP pour l'authentification, celle-ci ne peut reposer que sur le certificat X.509 qui doit absolument être correctement validé.

5 Implémentations

5.1 Côté serveur

Microsoft La partie historique (voir section 3) a présenté les différentes évolutions du protocole qui correspondent aux différentes versions et *Service Pack* de Windows. Cette partie présente des spécificités d'implémentation des serveurs Microsoft.

La figure 5 résume les différents niveaux disponibles côté serveur en *Standard RDP Security* en fonction des versions de Windows.

Niveau	2000	2003	2003 SP1	2008	XP	Vista / 7
Low	dispo	dispo	dispo	dispo	GPO	GPO
Client comp.	**défaut**	**défaut**	**défaut**	**défaut**	**défaut**	**défaut***
High	dispo	dispo	dispo	dispo	GPO	GPO
FIPS	N.D.	dispo	dispo	dispo	GPO globale	GPO globale

Légende :
- *dispo* : disponible et configurable ;
- *GPO* : disponible et configurable par GPO ;
- *GPO globale* : GPO influençant tout le système ;
- *N.D.* : non disponible.
- * : la documentation spécifie que le niveau par défaut est *High*, mais en pratique ce n'est pas le cas

Figure 5. Niveaux de sécurité, côté serveur, disponibles et configurés par défaut

La figure 6 résume les différents protocoles disponibles côté serveur en *Enhanced RDP Security* en fonction des versions de Windows.

Au niveau de la sécurité du mode *Standard RDP Security*, jusqu'à Windows 2003 inclus, la clé publique RSA utilisée pour le transport de la clé de session est en pratique limitée à 512 bits, la clé générée par le serveur étant systématiquement de cette taille. Il est donc impossible de garantir la sécurité des communications dans ce mode (voir section 4.1). Windows 2008 supporte des clés de taille supérieure mais propose néanmoins un mode de compatibilité permettant de repasser de 2048 bits à 512 bits,

Protocole	2000	2003	2003 SP1	2008	XP	Vista / 7
TLS	N.D.	N.D.	disponible	disponible	N.D.	disponible
CredSSP	N.D.	N.D.	N.D.	disponible	N.D.	disponible
par défaut	N.A	N.A.	standard	négocié	N.A	négocié

Légende :
- *disponible* : disponible et configurable (serveur) ;
- *standard* : *Standard RDP Security* ;
- *négocié* : négocié à la connexion ;
- *N.D.* : non disponible ;
- *N.A.* : non applicable.

Figure 6. Résumé de la disponibilité des protocoles *enhanced* et de la configuration par défaut

pour les clients (tels que Citrix) ne supportant pas les signatures trop grandes [36].

Lorsque TLS est utilisé, le choix des suites de chiffrement est primordial, notamment pour obtenir la propriété de PFS. Or, sous Windows, la configuration de cette option ne peut se faire qu'au niveau du système dans son intégralité. La partie "recommandations" présente la marche à suivre (voir section 6). Il reste de plus impossible de forcer l'approche directe (*direct approach*), la négociation initiale étant obligatoire. Mais le plus problématique reste l'impossibilité d'utiliser l'authentification mutuelle de TLS basée sur un certificat client. Bien que le protocole le supporte en théorie, l'interface de configuration ne permet pas une telle utilisation.

Lorsque CredSSP est configuré, il est impossible de forcer l'utilisation de NTLM ou de Kerberos.

Malgré l'utilisation massive de TLS (nativement ou par CredSSP, il n'est pas possible de réaliser l'authentification cliente *via* TLS.

Il est impossible de configurer les suites cryptographiques TLS utilisées par RDP (côté serveur ou client) sans affecter l'ensemble de la configuration du système. Cette modification n'est donc pas réaliste en pratique.

Autres implémentations En dehors de l'implémentation de Microsoft, il existe quelques implémentations de serveurs RDP. Bien qu'il soit difficile

36. KB949914 : `http://support.microsoft.com/kb/949914`

de les énumérer, car elles peuvent parfois être intégrées au sein d'équipements embarqués, on peut noter parmi les implémentations propriétaires celle de VirtualBox, disponible dans les plugins non-libres. Implémentée en C++, elle semble différente des autres implémentations connues mais implémente (au moins) TLS.

Du côté des implémentations libres, deux produits relativement proches implémentent le protocole : FreeRDP [2] et xrdp [11]. L'implémentation de FreeRDP reste expérimentale et propose d'exporter un serveur X *via* RDP. *xrdp* est un peu plus complexe et dispose d'un gestionnaire de session intégré lui permettant d'authentifier l'utilisateur localement mais également de faire passerelle vers d'autres serveurs RDP.

5.2 Côté client

Microsoft (*mstsc.exe*) Le client officiel de Microsoft est intéressant à étudier, son interface et son comportement changeant de manière significative au fil des versions. Comme présenté dans l'historique, le support de TLS n'apparaît qu'avec le client pour 2003 SP1. Le support de CredSSP arrive avec le client 6.1. Il reste néanmoins impossible de spécifier dans la configuration du client la couche de sécurité à utiliser. Le choix entre *Standard* et *Enhanced RDP Security* sera donc négocié à chaque connexion, ce qui pose évidemment un problème étant donné l'infériorité fondamentale du mode standard.

Standard RDP security :

Pour les clients utilisant le mode standard, il est impossible de choisir ou même d'obtenir des informations sur la taille de clé utilisée pour le chiffrement. Il en est de même pour l'utilisation des *salted MAC*.

La configuration du niveau FIPS ne peut être par ailleurs réalisée que par un paramètre [37] global du système (à partir de XP) ayant un impact sur l'**ensemble** des composants Windows utilisant de la cryptographie.

Il est donc impossible d'avoir une indication sur le niveau de sécurité en pratique : taille de la clé publique du serveur, algorithme et taille de clef utilisée ou encore utilisation de *salted MAC*. La seule "solution" est d'analyser les paquets réseau dans le détail.

Enhanced RDP Security (TLS et CredSSP) :

L'utilisation de CredSSP nécessite un client en version 6.1 au minimum. Lors de l'utilisation de ce moyen d'authentification, la logique de

37. `http://support.microsoft.com/kb/811833`

connexion et de validation de l'authentification du serveur par le client est relativement complexe. Le client, en version 7 et supérieure, considère le serveur comme authentifié lorsque :

- la connexion est établie en *Enhanced* **et** :
  - le certificat X.509 est validé par une chaîne de certification valide **ou**
  - l'authentification CredSSP s'effectue avec Kerberos.

Le client affiche alors un cadenas dans la barre d'état supérieure, indiquant le type d'authentification ayant été utilisée.

Contrairement à la version 7, la version 6.1 présente une vulnérabilité : le client considère qu'une authentification CredSSP avec NTLMSSP suffit à authentifier le serveur.

Les dernières versions du client Microsoft (versions 6.0 et supérieures) proposent en cas d'échec d'authentification de garder un condensat du certificat [38] dans le but de ne plus "importuner" l'utilisateur avec un message d'avertissement lors des prochaines connexions au même serveur, à la manière des clients SSH. Dans ce cas, la barre d'état après connexion ne montre pas de cadenas.

Tous les autres cas vont donc déclencher l'affichage d'un avertissement lorsque l'option de sécurité est positionnée à "**M'avertir**". Dans cette configuration, le client effectue les étapes suivantes :

1. établissement de la première connexion au serveur ne disposant pas d'un certificat valide ;
2. coupure de la connexion TCP ;
3. présentation du certificat serveur à l'utilisateur pour validation *ou* validation implicite si l'utilisateur a coché "ne plus m'avertir" ;
4. établissement d'une nouvelle session TCP ;
5. établissement du canal TLS ;
6. **ici, le client ne vérifie pas que le certificat présenté par le serveur correspond à celui présenté lors de la première connexion** ;
7. envoi du login / mot de passe protégé avec le mode de sécurité négocié.

Il est donc possible pour un attaquant pouvant réaliser un MITM d'obtenir le mot de passe de l'utilisateur en clair, sans que celui-ci puisse le détecter. Il lui suffit de laisser la première connexion TCP s'établir de manière transparente et d'effectuer une attaque MITM classique sur la

38. Le condensat du certificat est stocké dans l'attribut CertHash sous la clé `HKCU\Software\Microsoft\Terminal Server Client\Servers\Nom Serveur`.

deuxième connexion TLS. Néanmoins, une bonne configuration, décrite dans la partie 6, permet d'éviter ce problème.

Il est important de noter que même si le client disposait déjà de l'empreinte du certificat (si l'utilisateur l'avait au préalable sauvegardée) lors de la première connexion, celui-ci coupe toujours la connexion et ne valide pas le second certificat présenté.

Autres clients Les autres clients les plus connus sont bien évidemment *rdesktop* [4] et *FreeRDP* [2], le second étant un *fork* du premier, dont le développement stagnait quelque peu. Ils sont principalement utilisés pour se connecter depuis des machines non-Windows. *FreeRDP* évolue très rapidement et supporte depuis peu la majorité des fonctionnalités de la version 7 du protocole. Au niveau des options de sécurité, il supporte CredSSP et TLS.

> *rdesktop* ne supporte quasiment aucune fonction de sécurité : ni *salted MAC*, ni TLS, ni CredSSP. FreeRDP n'utilise pas les *salted MAC* par défaut et ne supporte pas Kerberos.

Il existe également de nombreux clients propriétaires, notamment pour plate-formes mobiles.

6 Recommandations

Cette partie présente l'ensemble des recommandations relatives aux aspects configuration et architecture permettant une utilisation plus sécurisée de RDP.

6.1 Configurer les options liées à RDP

Côté client :

Indépendamment du système d'exploitation, si le client Microsoft est utilisé (mstsc), la première recommandation est de le mettre à jour afin de disposer de la dernière version (actuellement la version 7.0). Ceci permet par exemple sur un Windows XP SP3 de bénéficier nativement de TLS pour protéger les sessions RDP. Ceci permet généralement de profiter d'amélioration d'ergonomie et de corrections de bogues. Toutefois, même avec un client à jour, l'ensemble des fonctionnalités supportées par la version du protocole RDP associée ne sont pas nécessairement disponibles automatiquement. C'est notamment le cas pour les mécanismes de sécurité. Par exemple, un client 7.0 sous Windows XP SP3 à jour ne

pourra pas s'appuyer sur CredSSP, sauf à activer spécifiquement celui-ci *via* la base de registre. Il est donc important de bien comprendre le niveau de sécurité atteint par la combinaison du client et du système d'exploitation : ceci demande un effort conséquent qui est en pratique difficilement réalisable du fait du volume de la documentation Microsoft.

Concernant les possibilités de paramétrage de la sécurité sur le client, celles-ci sont limitées. En pratique, l'interface permet uniquement de forcer l'authentification du serveur et de contrôler la réaction du client en cas d'échec d'authentification du serveur. L'option **Ne pas établir la connexion** permet ainsi :

1. de désactiver l'utilisation de *Standard RDP Security*, qui ne permet pas d'authentifier le serveur et autorise donc des attaques par le milieu ;
2. de stopper la connexion si l'authentification échoue.

Toutefois, même avec cette option activée, si NTLMSSP est utilisé par CredSSP pour l'authentification, le client avertira l'utilisateur et coupera la connexion uniquement **après** avoir transmis le défi/réponse (LM, NTLM ou NTLMv2).

Le client mstsc ne permet pas de configurer spécifiquement le SSP (Kerberos ou NTLMSSP) que CredSSP va utiliser.

Si CredSSP utilise NTLMSSP pour l'authentification et que celle-ci échoue, un défi/réponse a malgré tout été transmis.

On retombe alors sur des recommandations classiques liées à NTLMSSP, qui sont de disposer d'une politique de mots de passe forts et d'utiliser exclusivement NLTMv2. Ceci permet de se prémunir contre le cassage de mot de passe à partir d'un défi/réponse.

Dans un domaine Active Directory, l'option **Ne pas établir la connexion** peut être déployée *via* une stratégie de groupe[39]. Cette stratégie, une fois activée, empêche l'utilisateur d'ouvrir une session sur une machine non authentifiée par certificat (TLS) ou Kerberos (CredSSP).

Malgré tout, il est important de noter que le seul mode de protection des échanges disponibles sur un serveur RDP sous Windows XP est le mode *Standard RDP Security*. La recommandation concernant le comportement à adopter par le client empêche donc la connexion à ce type de système.

39. Configuration Ordinateur, Modèles d'administration, Composants Windows, Services Bureau à distance, Client Connexion Bureau à Distance, Configurer l'authentification du serveur pour le client.

Quant à CredSSP, il améliore le processus d'authentification mais son utilisation ne peut être forcée au niveau du client. Il est donc nécessaire d'éduquer les utilisateurs afin qu'ils puissent déterminer si CredSSP est utilisé (ou non) :

- si les authentifiants sont demandés avant la mise en place de la connexion et du déport d'affichage, CredSSP est utilisé ;
- si la connexion s'établit et que les authentifiants sont demandés dans l'affichage déporté, *via* les classiques *Credentials Providers*, CredSSP n'a pas été utilisé. Dans ce cas, il ne faut pas continuer la session (et surtout ne pas saisir ses authentifiants).

Le client mstsc ne permet pas de forcer l'utilisation de CredSSP ; il permet au mieux de désactiver *Standard RDP Security*.

Parmi les clients libres, FreeRDP est actuellement le seul fournissant un minimum de sécurité ; il est à utiliser avec les options `--sec tls` et *a minima* avec `--salted-checksum`.

Côté serveur :

Sur un système Windows serveur (2003 SP1 à 2008 R2), plusieurs options de configuration sont disponibles dans l'interface de configuration du service Terminal Server. Les paramètres recommandés sont les suivants :

- Couche de sécurité $\Longrightarrow$ SSL (TLS 1.0) : cette option permet de forcer le mode *Enhanced RDP Security* ;
- N'autoriser que la connexion des ordinateurs exécutant le Bureau à distance avec authentification NLA $\Longrightarrow$ activé : cette option permet d'imposer CredSSP pour l'authentification.

Ces paramètres peuvent être déployés au moyen d'une GPO [40].

6.2 Mettre en place une PKI

L'authentification du serveur lors de la montée de session TLS (validation du certificat serveur) est la seule solution d'authentification dans les environnements sans domaine Active Directory ou lorsque le client ne peut pas joindre un KDC (par exemple dans le cas d'accès distants).

Après la mise en place d'une PKI, un certificat X.509 doit être installé sur le serveur et sélectionné dans l'interface de configuration des services

40. Configuration Ordinateur, Modèles d'administration, Composants Windows, Services Bureau à distance, Hôte de la session à distance, Sécurité.

Terminal Server. Le *Common Name* de l'objet du certificat doit correspondre au nom utilisé par les clients pour référencer le serveur. De même, les attributs suivants doivent être positionnés dans le certificat :

- X509v3 Key Usage $\Longrightarrow$ Key Encipherment, Data Encipherment ;
- X509v3 Extended Key Usage $\Longrightarrow$ TLS Web Server Authentication

La configuration du client pour permettre la validation du certificat est également un prérequis à l'authentification du serveur. Pour cela, l'ancre racine de l'autorité doit être installée sur le client.

Malgré tout, dans un domaine où Kerberos est fonctionnel, CredSSP sera utilisé plutôt que TLS pour l'authentification du serveur. Dans ce cas, bien que CredSSP monte une session TLS, il n'imposera pas la validation du certificat présenté par le serveur, l'authentification du serveur pouvant intervenir ensuite *via* Kerberos. Dans ce cas, la mise en place d'une PKI n'apporte malheureusement rien.

6.3 Utiliser IPsec

IPsec, nativement intégré aux systèmes Windows depuis la version 2000, peut être utilisé pour protéger le trafic réseau lié à RDP. L'utilisation d'IPsec permet l'authentification mutuelle par certificat du client et du serveur lors de la négociation IKE initiale.

La mise en œuvre d'une telle architecture nécessite néanmoins la création d'une PKI pour supporter la génération des certificats client et serveur.

Sur un domaine Active Directory, l'utilisation de Kerberos pour l'authentification mutuelle lors de la négociation IKE pourra se substituer à l'utilisation de certificats. Malgré tout, ce mode nécessite l'accès au KDC et limite donc son déploiement au réseau local.

Même si les garanties obtenues sont fortes, la mise en œuvre d'IPsec dans les environnements Windows a un coût élevé.

6.4 Changer les pratiques d'administration

Un autre moyen pour sécuriser RDP est de ne pas l'utiliser.

De manière générale, les administrateurs utilisent RDP afin d'ouvrir une session graphique sur un serveur dans le but d'utiliser les outils d'administration installés localement sur le serveur.

Or certains services peuvent être administrés à distance et pas nécessairement depuis le serveur sur lequel ils s'exécutent. Dans ce cas, la

méthode consiste à utiliser des outils d'administration depuis un système distant dédié.

C'est en particulier le cas pour les services Microsoft dont la quasi-totalité sont administrables à distance, généralement au moyen des RPC. De même, les outils baptisés « Outils d'administration de serveur distant » permettent d'administrer tous les rôles et les fonctionnalités d'un serveur Windows (contrôleur de domaine, gestion des utilisateurs, DNS, DHCP, etc.).

Microsoft pousse d'ailleurs de plus en plus vers cette méthode d'administration à distance avec les versions dénommées « Server Core » des éditons serveur de Windows qui sont dépourvues d'interface graphique.

Enfin, Powershell prend une part de plus en plus importante dans l'administration des services Microsoft *via* des services Web de type WinRM [41] ou ADWS [42].

Néanmoins, ces alternatives se limitent généralement aux services Windows et ne permettent donc pas nativement de supporter les besoins d'administration d'applications tierces. De plus, même si ceux-ci ne nécessite pas le transport du mode de passe ou du code PIN de l'utilisateur, leur sécurité reste à étudier, ceux-ci ayant généralement été développés bien avant RDP et pour une utilisation sur le réseau local, supposé de confiance.

6.5 Architecturer le réseau

Au final, si RDP doit être utilisé, il faut le considérer comme un protocole d'administration et donc l'isoler des flux clients au moyen d'une architecture réseau adapté.

Ainsi, des réseaux d'administration dédiés doivent être créés, séparés des réseaux utilisateurs. Ceci peut être réalisé

- par de la segmentation au niveau 2 (VLAN ou commutateurs dédiés)
- au niveau 3 en plaçant les réseaux d'administration sur des plans d'adressage spécifiques ; l'accès aux serveurs à administrer étant alors routé et filtré pour n'autoriser que les machines d'administration à accéder au service RDP.

Dans un environnement Active Directory, la segmentation au niveau 2 trouve rapidement ces limites, notamment du fait que la seule configuration supporté pour un DC se limite à une interface réseau unique. Elle reste néanmoins utilisable pour les autres serveurs.

41. Windows Remote Management.
42. Active Directory Web Services.

7 Conclusion

D'un point de vue fonctionnel, RDP est un protocole bien architecturé comme le montrent les nouvelles fonctionnalités ajoutées au fil des versions. Il est également particulièrement efficace en matière de réactivité et de bande passante. En revanche, ces avantages sont obtenus au prix d'une complexité protocolaire qui rend sa compréhension et son implémentation difficiles.

De plus, les mécanismes de sécurité *ad-hoc* intégrés initialement se sont avérés vulnérables à de nombreuses attaques élémentaires. Les corrections et évolutions apportées depuis ont permis d'améliorer sensiblement le niveau de sécurité, sous réserve de configuration.

Néanmoins les points suivants viennent ternir le tableau :

- pour supporter les besoins de rétro-compatibilité, l'architecture actuelle des mécanismes de sécurité est extrêmement complexe ;
- de par leur conception et malgré l'utilisation sous-jacente de protocoles éprouvés (TLS), les nouveaux mécanismes propriétaires restent difficiles à évaluer ;
- les implémentations de Microsoft ne permettent ni la configuration ni la visualisation des paramètres de sécurité.

Au final, si RDP doit être utilisé, il est nécessaire de mettre en œuvre les recommandations présentées en section 6 pour atteindre un niveau de sécurité acceptable.

Références

1. Blog de l'équipe Remote Desktop Services. http://blogs.msdn.com/b/rds/.
2. FreeRDP. http://www.freerdp.com/.
3. Microsoft Windows NT Server, Terminal Server Edition, version 4.0 : An Architectural Overview. http://technet.microsoft.com/en-us/library/cc751283.aspx.
4. rdesktop : A Remote Desktop Protocol client. http://www.rdesktop.org/.
5. Remote Desktop Services Component Architecture Poster. http://www.microsoft.com/download/en/details.aspx?id=3262.
6. Référentiel Général de Sécurité. http://www.ssi.gouv.fr/fr/reglementation-ssi/referentiel-general-de-securite/.
7. Various RDP-related articles on MS Technet. http://technet.microsoft.com/en-en/.
8. Wikipedia Article : Remote Desktop Protocol. http://en.wikipedia.org/wiki/Remote_Desktop_Protocol.
9. Wikipedia Article : Remote Desktop Services. http://en.wikipedia.org/wiki/Remote_Desktop_Services.
10. Windows 2000 High Encryption Pack. http://www.microsoft.com/download/en/details.aspx?id=15667.
11. xrdp. http://www.xrdp.org/.
12. FIPS 140-1 : Security requirements for cryptographic modules. http://csrc.nist.gov/publications/fips/fips1401.htm, 01 1994.
13. Microsoft KB275727 : High Encryption on a Remote Desktop or Terminal Services Session Does Not Encrypt All Information. http://support.microsoft.com/default.aspx?scid=kb;en-us;275727, 2001.
14. Microsoft Security Bulletin MS01-006 : Invalid RDP Data can cause Terminal Server Failure. http://technet.microsoft.com/security/bulletin/MS01-006/ et http://support.microsoft.com/default.aspx?scid=kb;en-us;286132, 01 2001.
15. Microsoft Security Bulletin MS01-040 : Invalid RDP Data can Cause Memory Leak in Terminal Services. http://technet.microsoft.com/security/bulletin/MS01-040/ et http://support.microsoft.com/default.aspx?scid=kb;en-us;292435, 07 2001.
16. Microsoft Security Bulletin MS01-052 : Invalid RDP Data can Cause Terminal Service Failure. http://technet.microsoft.com/security/bulletin/MS01-052/ et http://support.microsoft.com/default.aspx?scid=kb;en-us;307454, 10 2001.
17. Microsoft Security Bulletin MS02-046 : Buffer Overrun in TSAC ActiveX Control Could Allow Code Execution. http://technet.microsoft.com/security/bulletin/MS02-046/ et http://support.microsoft.com/default.aspx?scid=kb;en-us;327521, 08 2002.
18. Microsoft Security Bulletin MS02-051 : Cryptographic Flaw in RDP Protocol can Lead to Information Disclosure. http://technet.microsoft.com/security/bulletin/MS02-051/ et http://support.microsoft.com/default.aspx?scid=kb;en-us;324380, 09 2002.

19. Factorisation de clef RSA 518 bits. http://www.mersenneforum.org/showpost.php?p=124079&postcount=97, 2003.
20. Microsoft Terminal Services vulnerable to MITM-attacks. http://www.securityfocus.com/archive/1/317244, May 2003.
21. CVE-2005-1794. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-1794 et http://www.oxid.it/downloads/rdp-gbu.pdf, 05 2005.
22. Microsoft Security Bulletin MS05-041 : Vulnerability in Remote Desktop Protocol Could Allow Denial of Service. http://technet.microsoft.com/security/bulletin/MS05-041/ et http://support.microsoft.com/default.aspx?scid=kb;en-us;899591, 08 2005.
23. Microsoft Security Bulletin MS09-044 : Vulnerabilities in Remote Desktop Connection Could Allow Remote Code Execution. http://technet.microsoft.com/security/bulletin/MS09-044/ et http://support.microsoft.com/default.aspx?scid=kb;en-us;970927, 08 2009.
24. Credential Security Support Provider (CredSSP) Protocol Specification. http://microsoft.com/, 09 2011.
25. Microsoft Security Bulletin MS11-017 : Vulnerability in Remote Desktop Client Could Allow Remote Code Execution. http://technet.microsoft.com/security/bulletin/MS11-017/ et http://support.microsoft.com/default.aspx?scid=kb;en-us;2508062, 03 2011.
26. Microsoft Security Bulletin MS11-061 : Vulnerability in Remote Desktop Web Access Could Allow Elevation of Privilege. http://technet.microsoft.com/security/bulletin/MS11-061/ et http://support.microsoft.com/default.aspx?scid=kb;en-us;2546250, 08 2011.
27. Microsoft Security Bulletin MS11-065 : Vulnerability in Remote Desktop Protocol Could Allow Denial of Service. http://technet.microsoft.com/security/bulletin/MS11-065/ et http://support.microsoft.com/default.aspx?scid=kb;en-us;2570222, 08 2011.
28. Remote Desktop Protocol : Basic Connectivity and Graphics Remoting Specification. http://microsoft.com/, 03 2011.
29. Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) Extension. http://microsoft.com/, 09 2011.
30. Microsoft Security Bulletin MS12-020 : Vulnerabilities in Remote Desktop Could Allow Remote Code Execution. http://technet.microsoft.com/security/bulletin/MS12-020/ et http://support.microsoft.com/default.aspx?scid=kb;en-us;2671387, 03 2012.
31. Hugo Krawczyk. The order of encryption and authentication for protecting communications (or : how secure is ssl?). *IACR Cryptology ePrint Archive*, 2001 :45, 2001.
32. G. Pall. Microsoft Point-To-Point Compression (MPPC) Protocol. RFC 2118 (Informational), March 1997.