Diode réseau et ExeFilter : 2 projets pour des interconnexions sécurisées

Philippe Lagadec

DGA / CELAR
philippe.lagadec(à)dga.defense.gouv.fr

Résumé La Diode réseau et ExeFilter sont deux projets complémentaires du CELAR pour construire des interconnexions hautement sécurisées entre réseaux de niveaux de sensibilité différents. Ils visent à permettre par exemple l'interconnexion de réseaux sensibles avec Internet, en garantissant une prise de risque minimale vis-à-vis des chevaux de Troie et autres menaces.

La Diode réseau est conçue pour garantir un transfert de données dans un seul sens, en se basant sur une liaison optique unidirectionnelle. Une application simple est par exemple le téléchargement automatique de mises à jour antivirus vers un réseau sensible, tout en empêchant la moindre fuite de données vers Internet.

ExeFilter est un filtre générique de fichiers ou de courriels, qui permet de s'assurer que seuls certains formats de fichiers maîtrisés sont acceptés, et que ceux-ci ne contiennent aucun contenu actif. (en se basant sur les principes énoncés lors de [SSTIC03] et [SSTIC04])

1 Introduction

Les produits de sécurité actuellement disponibles sur le marché (pare-feux, proxies applicatifs, antivirus, IDS, IPS, ...) se révèlent indispensables mais souvent insuffisants pour assurer un haut niveau de confiance lorsqu'il s'agit d'interconnecter un réseau sensible et un réseau non maîtrisé comme Internet. En effet, il est très difficile de se protéger contre des chevaux de Troie spécifiquement développés et correctement camouflés dans des flux applicatifs. De plus, la plupart des protocoles standards offrent de nombreux canaux cachés permettant à des chevaux de Troie ou des utilisateurs malveillants d'exporter des données sensibles vers l'extérieur.

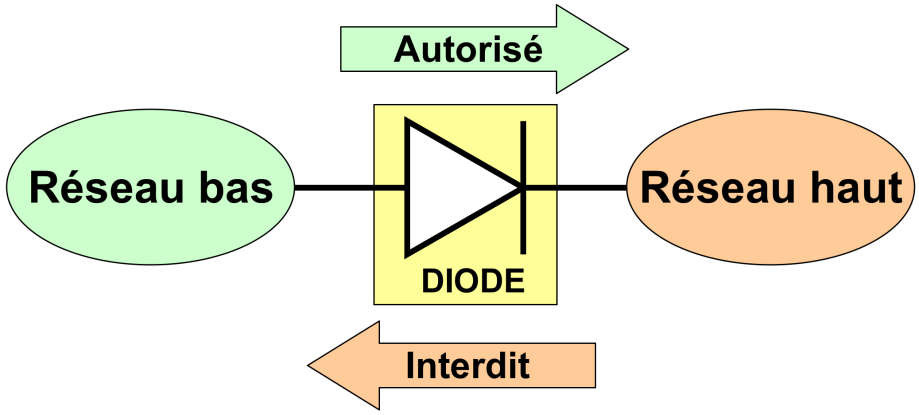
Pour répondre aux besoins d'interconnexion de réseaux sensibles avec des réseaux non maîtrisés, le CELAR a conçu et développé 2 solutions techniques complémentaires pour améliorer le niveau de confiance des passerelles de filtrage :

- la **Diode réseau** pour bâtir des interconnexions unidirectionnelles afin de transférer des données d'un réseau de « niveau bas » vers un réseau de « niveau haut », sans risque de fuite dans l'autre sens à travers un canal caché.
- **ExeFilter** pour filtrer les fichiers et courriels suivant une politique stricte, afin de se prémunir contre l'entrée de contenu actif pouvant camoufler un cheval de Troie ou tout code malveillant (y compris tous les virus).

Cet article présente ces deux projets qui ont été développés en langage Python, et qui sont actuellement à l'état de prototypes utilisés en semi-production depuis mi-2005.

2 Diode réseau optique

Une diode réseau est un système qui permet d'interconnecter 2 réseaux, en autorisant le transfert de données **dans un seul sens**. Ce type de système est généralement employé pour relier un réseau nécessitant un niveau de sécurité élevé, appelé « réseau haut » à un réseau de confiance moindre (par exemple Internet), le « réseau bas ». Seul la remontée d'informations du réseau bas vers le haut est autorisée, afin de garantir la confidentialité du réseau haut.



Cet article présente le projet Diode réseau du CELAR, qui a pour but de montrer qu'il est possible de mettre en place une liaison diode à laide de matériel standard avec un coût d'acquisition faible, tout en apportant un niveau de confiance élevé.

2.1 Historique du projet

Bien sûr, l'idée d'une « diode réseau » n'est pas neuve. Il est facile de trouver sur Internet des publications de travaux de recherche dans ce domaine qui remontent aux années 90 voire 80 (cf. [DSTO1,DSTO2,NPUMP]), s'appuyant sur des technologies optiques ou sur des liaisons série RS-232. Certaines sociétés proposent même des produits « diode » complets sur étagère, comme [TENIX].

On peut cependant remarquer que la plupart des recherches et des applications sont pour l'instant étroitement liées à des besoins militaires. Cela peut s'expliquer par le fait que les réglementations militaires divisent souvent les réseaux en différents niveaux de sensibilité, et imposent un cloisonnement très strict entre ces niveaux.

D'autre part à notre connaissance ce type de produit n'est toujours pas disponible en France, ce qui a fortement motivé le lancement de notre projet, ainsi que cette participation au symposium SSTIC.

Les travaux concrets du CELAR sur la Diode réseau optique ont débuté fin 2001. Un premier prototype a été développé en langage C en 2002, utilisé en plate-forme et amélioré jusqu'en 2005. Cette première version offrait des services de transfert de fichiers, de synchronisation de répertoires, et de transfert de messagerie Sendmail. Le projet présenté dans cette article est un second prototype réécrit en langage Python depuis 2005, afin de simplifier le code source et d'apporter de nouvelles améliorations.

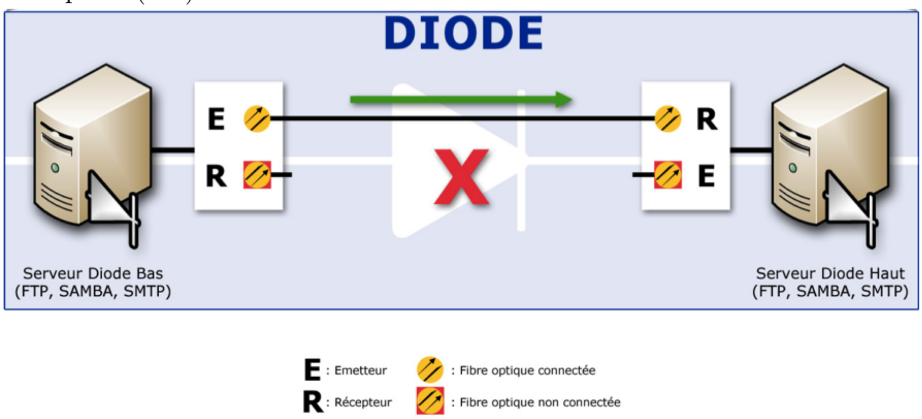
2.2 Vue d'ensemble

La version actuelle de la Diode réseau du CELAR est volontairement très simple : elle permet uniquement la recopie automatisée de fichiers ou d'arborescences complètes de répertoires d'un serveur bas vers un serveur haut. Cela permet déjà de mettre en oeuvre de nombreuses applications, que nous allons détailler plus loin. Il est prévu d'améliorer à moyen terme cette version pour élargir les services apportés, notamment le transfert de messagerie et de flux de supervision.

La diode réseau est constituée de 2 parties, matérielle et logicielle, décrites dans les paragraphes suivants.

2.3 Partie matérielle

La partie matérielle de la diode est une liaison Ethernet optique unidirectionnelle qui relie 2 serveurs. Il s'agit de matériel standard peu onéreux. Le caractère unidirectionnel de la liaison est garanti par l'utilisation d'une seule fibre optique reliée du côté « bas » à un port émetteur optique (TX), du côté « haut » à un port récepteur (RX).



Les caractéristiques optiques et électroniques intrinsèques du matériel assurent qu'aucune donnée ne peut être transmise du haut vers le bas, il n'existe donc pas de canal caché possible. C'est cette propriété « matérielle » qui distingue fondamentalement la diode réseau d'un pare-feu classique : Même s'il est possible de configurer un pare-feu pour bâtir une liaison unidirectionnelle

entre deux réseaux, on ne pourra jamais assurer qu'une vulnérabilité logicielle ou un canal caché ne puisse pas un jour permettre de transférer des données dans l'autre sens.

Il existe au moins 2 possibilités pour obtenir une telle liaison optique unidirectionnelle :

- Installer une carte réseau Ethernet optique dans chacun des serveurs.
- Utiliser des cartes Ethernet classiques (RJ45) dans les serveurs, et insérer 2 boîtiers convertisseurs (transceivers) RJ45/optique entre les serveurs (cf. schéma ci-dessus).

Ces 2 possibilités aboutissent strictement au même résultat, cependant nous avons privilégié la seconde, car cela permet de facilement caractériser la liaison diode en tant que matériel indépendant de tout PC et de tout logiciel. Chaque serveur dispose également d'une autre carte réseau, afin de le connecter au réseau bas et haut.

Problème de réception Le principal problème rencontré lors de la mise en place de cette partie matérielle est lié au fait que la plupart des cartes et des convertisseurs optiques actuels vérifient « intelligemment » l'état de la connexion pour fonctionner. Ils nécessitent donc la réception régulière d'un signal sur leur port de réception RX, ce qui n'est pas le cas si on se contente simplement de débrancher une des 2 fibres optiques comme indiqué sur le schéma ci-dessus.

Pour contourner ce problème, nous avons testé ou envisagé diverses solutions :

- Acheter un matériel qui ne fait pas cette vérification, ou qui peut être configuré pour ne pas le faire. Ce type de matériel semble hélas de plus en plus difficile (voire impossible) à trouver.
- Modifier légèrement l'électronique interne du convertisseur pour désactiver ou leurrer cette fonction (soudure d'une simple résistance judicieusement placée, par exemple). Cela risque cependant de compromettre la garantie du matériel, voire de le rendre inutilisable.
- Connecter un troisième convertisseur branché « dans le vide » sur ce port RX (cf. [DSTO1,DSTO2] pour de beaux schémas). D'après nos quelques essais cela fonctionne avec la plupart des modèles, seuls certains posent problème.
- Concevoir un système optique capable de reboucler une partie du signal émis du port TX sur le port RX, ou de simuler ce signal. Cette solution plus élégante s'éloigne cependant de notre objectif de départ qui était d'utiliser uniquement du matériel standard.

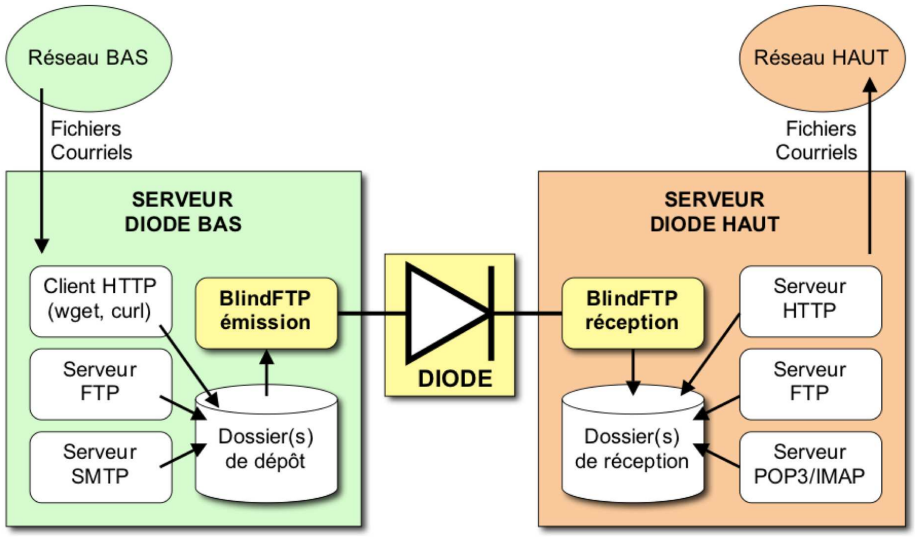
Au final il est donc indispensable de bien étudier le matériel acheté pour concevoir une diode réseau.

2.4 Partie logicielle - BlindFTP

La partie logicielle de la diode réseau correspond à 2 processus, chacun étant installé sur un des serveurs. La partie cliente sur le serveur « bas » est chargée d'émettre les fichiers d'un répertoire à travers la liaison optique, en utilisant le

protocole UDP. La partie serveur sur le serveur « haut » reçoit les tronçons de fichiers et les reconstitue dans un répertoire destination. L'ensemble a pour but de synchroniser un répertoire du serveur bas à l'identique sur le serveur haut.

Ce fonctionnement simple permet de transférer facilement des fichiers ou des courriels reçus par d'autres clients ou serveurs HTTP, FTP, SMTP ou autres, afin de fournir des services aux réseaux haut et bas.



Protocole BlindFTP Le principal problème est que le protocole UDP ne garantit ni l'acheminement des données à destination, ni leur intégrité. Seul UDP est utilisable puisque la liaison unidirectionnelle ne permet aucun acquittement, TCP ne peut fonctionner sur une liaison de ce type. La transmission de données se fait donc en aveugle.

Selon la charge CPU du serveur haut, on constate que le tampon de la pile IP est régulièrement saturé si le processus de réception ne traite pas les données suffisamment vite. Certains datagrammes UDP sont alors perdus. Cette situation se produit notamment lorsque le processus est occupé à écrire les données reçues sur le disque dur.

Pour assurer une transmission complète et intègre des fichiers, nous avons donc dû développer un (modeste) protocole de transfert de fichiers spécifique en aveugle sur UDP, baptisé « BlindFTP ». Il assure tout d'abord une limitation du débit d'émission, afin de pouvoir s'adapter aux performances des serveurs employés. Il est également basé sur la redondance de l'envoi : chaque fichier est transmis N fois, et le processus de réception complète chaque fois les tronçons de fichiers manquants.

La première version de la diode assurait une redondance au niveau de chaque datagramme afin de garantir l'intégrité d'un flux quelconque (à la manière de TCP), ce qui a abouti au final à un fonctionnement relativement complexe et à un code source difficile à maintenir et améliorer.

A l'inverse, BlindFTP est conçu de la façon la plus simple possible, et les performances obtenues sont aussi satisfaisantes :

Côté émission, chaque fichier à transmettre est divisé en blocs de la taille choisie d'un datagramme UDP (par exemple 1400 octets pour éviter la fragmentation) moins la taille de l'entête BlindFTP. Ensuite, on ajoute à chaque bloc une entête qui contient les informations suivantes :

- Nom, taille et date du fichier
- Numéro du bloc dans le fichier, et nombre total de blocs
- Index du début du bloc dans le fichier (offset)
- Checksum CRC32 du fichier

Ces informations sont ajoutées dans chaque bloc, ce qui constitue une certaine redondance et un certain overhead. Cependant cela permet de simplifier le processus de réception, car n'importe quel datagramme peut être perdu. Même si un seul bloc d'un fichier est reçu, les informations nécessaires à son traitement sont disponibles.

Le nom de fichier contient le chemin relatif du fichier par rapport au répertoire source.

Côté réception, le processus BlindFTP reconstitue progressivement chaque fichier émis dans un fichier temporaire, en notant les numéros de blocs reçus, et en comblant les trous si des blocs ont été « perdus ». Dès que tous les blocs d'un fichier sont reçus, celui-ci est recopié à destination. On vérifie son intégrité à laide de l'empreinte CRC32, et son nom est analysé pour éviter les vulnérabilités de type « directory traversal ».

Quand une perte de certains datagrammes se produit, plusieurs transmissions du même fichier sont nécessaires pour compléter l'ensemble des blocs. Le processus d'émission est donc configuré l'adresse pour renvoyer chaque fichier de l'arborescence source soit indéfiniment, soit un nombre jugé suffisant de fois (typiquement 10 ou 20), afin de réduire le risque de ne pas recevoir un fichier.

Il aurait été possible d'employer un protocole existant pour assurer le transfert de fichiers sur une liaison unidirectionnelle, comme par exemple [FLUTE]. Ce type de protocole n'est cependant pas adapté à une simple liaison diode en raison de sa complexité, car il comprend des fonctionnalités destinées à la diffusion sur Internet via multicast, avec contrôle de flux. Un protocole plus simple comme BlindFTP permet de réduire au strict minimum la charge du serveur qui réceptionne les données, ce qui limite les pertes de datagrammes et améliore donc les performances.

Configuration L'utilisation du protocole UDP nécessite sur le serveur bas la conversion de l'adresse IP du serveur haut en adresse MAC. A cause de la liaison diode, le protocole ARP normalement chargé de cette conversion n'est pas utilisable. Il est donc nécessaire de configurer statiquement le couple d'adresses

MAC/IP du serveur haut dans la table ARP du serveur bas, ce qui peut se faire par exemple grâce à la commande arp.

Une autre solution aurait pu être d'utiliser une adresse de broadcast IP convertie en broadcast Ethernet, ou bien à forger directement des trames Ethernet avec l'adresse du serveur haut.

Performances Sans optimisation particulière avec un programme en langage Python relativement simple tournant sur des PCs standards à 1GHz, il est possible d'obtenir un débit moyen de 12 Mbps à partir d'une liaison Ethernet optique à 100 Mbps.

Des travaux en cours devraient permettre d'améliorer ces performances, notamment :

- Utilisation de Psyco (compilateur JIT pour Python, cf. [PSYCO])
- Optimisation des paramètres système de la pile IP
- Multithreading
- Utilisation intensive de la mémoire vive comme tampon
- Utilisation de redondance et de parité (type PAR2, cf. [PAR2])
- Diverses stratégies d'ordonnancement pour l'envoi des fichiers

2.5 Applications

De nombreuses applications sont possible à partir d'une diode réseau et du protocole BlindFTP :

Transfert de fichiers manuel Un répertoire de dépôt est accessible aux utilisateurs du réseau bas par FTP ou un partage Windows/Samba. Il est synchronisé vers un répertoire du réseau haut, lui aussi accessible aux utilisateurs grâce à un autre serveur (cf. schéma ci-dessus).

Transfert de fichiers automatisé - mises à jour antivirus Cela peut être par exemple employé pour des mises à jour de signatures antivirus ou d'outils comme Nmap, Nessus, MBSA, etc.

Un client web comme wget ou curl est utilisé dans une tâche planifiée pour télécharger régulièrement la nouvelle version de fichiers de mise à jour disponibles sur Internet. Ces fichiers sont ensuite recopiés via la diode sur le réseau haut, afin de mettre à jour les outils correspondants.

Ce type d'application élimine les fastidieuses mises à jour manuelles par supports amovibles, et permet d'améliorer la réactivité en augmentant la fréquence des mises à jour.

Recopie de sites web En exploitant les fonctions d'aspiration de sites web d'outils comme wget ou curl, il est envisageable de constituer un miroir de sites web sur le réseau haut à travers la diode. Dans ce cas il est utile de combiner la diode avec un filtrage de contenu comme ExeFilter pour se protéger contre d'éventuels codes malveillants.

Transfert de mises à jour Windows avec WSUS WSUS est la solution de Microsoft qui permet de constituer un serveur « Windows Update » à l'intérieur d'un réseau d'entreprise, afin d'éviter que chaque poste client doive se connecter à Internet pour bénéficier des mises à jour automatiques. L'inconvénient majeur de cette solution est que le serveur WSUS doit obligatoirement être connecté à Internet pour télécharger régulièrement les correctifs.

Depuis mi-2005, la nouvelle version du serveur WSUS a apporté une fonction révolutionnaire, qui permet enfin de transférer la base de données d'un serveur WSUS vers un autre par simple recopie de fichiers. Le serveur WSUS connecté au réseau d'entreprise n'a donc plus besoin d'être connecté à Internet.

Nous utilisons cette solution depuis plusieurs mois avec succès à travers la diode CELAR, ce qui permet d'optimiser la mise à jour des postes clients et serveurs Windows tout en garantissant la confidentialité du réseau connecté à la diode.

Transfert de mises à jour Linux Comme la plupart des distributions Linux proposent également des systèmes de mises à jour automatisées, il est envisageable de fournir ce service à travers une liaison diode. C'est une des applications que nous avons prévu de développer à court terme.

Transfert de messagerie SMTP Il est possible de synchroniser une file d'attente de messages d'un serveur (par exemple Sendmail ou Postfix), afin de recevoir sur le réseau haut des messages émis depuis le réseau bas. En effet la plupart des serveurs sauvegardent les messages reçus comme des fichiers dans un simple répertoire, qu'il est possible de recopier vers un autre serveur.

Cette application avait été mise en oeuvre avec la première version de la diode, mais n'a pas encore été adaptée à la version actuelle.

Synchronisation de bases de données, d'annuaires, ... Comme cela est le cas avec WSUS, n'importe quelle base de données est exportable et importable sous forme de fichiers. Une liaison diode peut donc servir à recopier une base de données quelconque d'un réseau bas vers un réseau haut. Il serait même envisageable d'optimiser le processus en adaptant le protocole BlindFTP pour fonctionner directement avec les serveurs de bases de données sans passer par des imports/exports de fichiers.

Archivage sécurisé Certains systèmes d'information hautement sécurisés nécessitent la protection à long terme de l'intégrité de certaines données archivées, par exemple de journaux de sécurité afin de garantir l'imputabilité des actions et la non-répudiation. Une diode peut être utilisée pour recopier régulièrement ces informations vers un réseau « sanctuaire » où les utilisateurs ne peuvent se connecter pour les effacer ou les modifier après transfert.

Supervision réseau et sécurité Une liaison diode est également exploitable pour des protocoles plus simples basés sur UDP. Il est par exemple envisageable de transmettre directement des protocoles unidirectionnels comme syslog ou SNMP-trap afin de permettre la supervision du réseau bas par la réseau haut.

Détection d'intrusion furtive - Honeypots Pour finir, une liaison optique unidirectionnelle permet également l'écoute réseau, par exemple pour la détection d'intrusion. Pour cela, il suffit de remplacer le « serveur diode bas » sur les schémas ci-dessus par un concentrateur réseau (hub) ou par un commutateur (switch) avec un port configuré pour relayer tout le trafic. Le serveur diode haut héberge quant à lui le logiciel de détection d'intrusion. Celui-ci peut alors écouter tout le trafic, mais il ne peut rien émettre, ce qui le rend donc totalement furtif même en cas de vulnérabilité.

Ce type de solution pourrait aussi être intéressante dans le cadre des pots de miel, pour remonter des informations de façon plus furtive.

3 ExeFilter

ExeFilter est un logiciel qui permet d'analyser et de filtrer des fichiers ou des courriels, afin de ne laisser passer que des contenus maîtrisés, statiques et inoffensifs. Tout contenu actif comme un fichier exécutable, une macro ou un script peut être bloqué ou nettoyé (cf. [SSTIC03,SSTIC04]).

Une politique de filtrage stricte est appliquée, pour n'accepter que certains formats de fichiers connus et maîtrisés. L'analyse est récursive si des formats conteneurs sont détectés (archives ZIP, TAR, Gzip,) Ce module est conçu de façon générique, afin de pouvoir l'employer pour divers protocoles : FTP, SMTP, HTTP, ...

Aujourd'hui des filtres ont été créés pour analyser les formats les plus usuels : TXT, HTML, PDF, DOC, XLS, PPT, RTF, JPG, GIF, MP3, AVI, ZIP, ...

3.1 Objectifs

L'objectif principal d'ExeFilter est de protéger l'intégrité d'un système d'information particulièrement sensible, lorsque celui-ci reçoit des fichiers ou des courriels provenant d'autres systèmes d'un niveau de confiance inférieur (par exemple Internet). Cette intégrité est particulièrement menacée par les chevaux de Troie, virus et autres codes malveillants contenus dans ces fichiers et courriels. Comme cela avait été rappelé lors de [SSTIC03], il est impossible de concevoir un filtre capable de discriminer sans erreur un code exécutable « inoffensif » d'un code malveillant quelconque (connu ou inconnu).

Pour de nombreux systèmes d'information nécessitant un haut niveau de sécurité, les outils de protection classiques comme les antivirus ou les filtres basés sur des méthodes heuristiques ne sont pas suffisants. En effet leur conception suppose que l'on accepte toujours le risque de laisser passer certains codes malveillants inconnus. Une solution plus satisfaisante est donc de filtrer toute

forme de code exécutable, afin de n'accepter que des contenus totalement statiques et inoffensifs. C'est l'objectif d'ExeFilter.

Il est à noter qu'ExeFilter protège contre quasiment tous les virus connus, puisque ceux-ci se répliquent sous une forme de fichiers exécutables qui n'est pas acceptée par ExeFilter dans une configuration « standard » (cf. [VBPREV] pour un aperçu des statistiques de virus actifs actuellement).

3.2 Historique du projet

Le CELAR se penche sur le thème de l'analyse de contenu (fichiers et courriels) depuis au moins 2000. Après plusieurs projets divers et variés dans ce domaine, basés sur l'intégration d'outils commerciaux, nous avons abouti au constat qu'il n'est pas encore possible d'obtenir un filtrage satisfaisant à l'aide d'outils disponibles sur le marché.

Le projet qui a mené à ExeFilter a débuté fin 2004. Un premier développement très rapide en langage Python a permis de montrer qu'il était possible de concevoir un filtre très efficace en se basant sur un algorithme solide, décrit ci-après. Aujourd'hui ce projet est utilisé en semi-production depuis fin 2005, et les résultats sont très encourageants.

3.3 Limites et contraintes

Même si ExeFilter permet d'aboutir à un niveau de confiance beaucoup plus élevé qu'un antivirus classique, une protection absolue est impossible.

Tout d'abord, un filtre tel qu'ExeFilter est obligatoirement limité à un nombre fini de formats de fichiers connus et maîtrisés. Il est indispensable de connaître parfaitement un format pour pouvoir identifier quels sont les contenus inoffensifs et quels sont les contenus actifs.

De plus l'efficacité d'un tel filtrage ne peut être garantie à 100% au cours du temps :

- Les formats de fichiers évoluent. Un format maîtrisé et connu à un instant donné peut être enrichi et apporter de nouvelles fonctionnalités problématiques en terme de sécurité quelques mois plus tard.
- L'interprétation d'un format de données dépend fortement de l'application qui ouvre le fichier, et ces applications peuvent évoluer. Ainsi un même fichier HTML inoffensif pour Mozilla Firefox peut se révéler malveillant lorsqu'il est ouvert dans Internet Explorer (et vice-versa).
- L'exploitation d'une vulnérabilité d'implémentation d'une application, permettant par exemple l'exécution de code arbitraire à l'aide d'un débordement de tampon. Cette vulnérabilité n'est pas liée au format de fichier lui-même, mais à l'application.

Un filtrage strict tel que celui appliqué par ExeFilter est très contraignant pour l'utilisateur, car il interdit de nombreuses fonctionnalités « actives » habituellement employées pour agrémenter les documents : macros, scripts, animations, ... Il ne peut donc être mis en oeuvre que pour des interconnexions nécessitant un

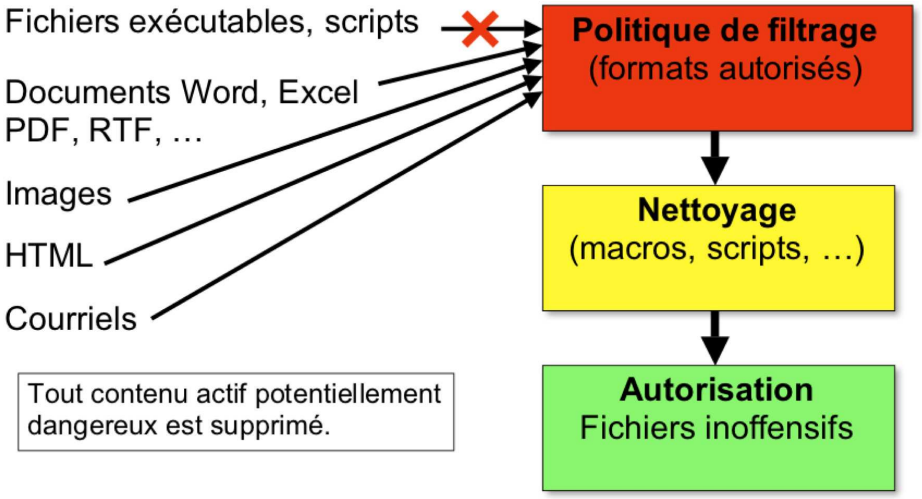
niveau de confiance particulièrement élevé. Les besoins constants d'échanger de nouveaux formats de fichiers posent notamment problème, car il est nécessaire de mener une analyse de risques sur chaque format de fichier.

Au final ExeFilter permet de réduire drastiquement le risque de transmettre un cheval de Troie ou un virus camouflé dans un fichier ou un courriel, au prix d'une réduction notable de certaines fonctionnalités.

3.4 Algorithme

Le fonctionnement d'ExeFilter repose sur quelques idées simples :

- On ne veut autoriser que des formats de fichiers ou de courriels connus et maîtrisés. Pour cela, un fichier doit être obligatoirement reconnu par son nom (extension) ET par son contenu.
- Les contenus autorisés doivent être inoffensifs pour le système d'information qui les reçoit. On doit les « nettoyer » si cela est possible.
- Les fichiers inclus dans un format conteneur doivent être récursivement analysés.



Avant de décrire ce fonctionnement, il est nécessaire de bien définir quelques termes :

Dépollution ou nettoyage Action qui consiste à supprimer ou désactiver tout contenu interne d'un fichier (ou d'un ensemble de fichiers) susceptible de poser des problèmes de sécurité (conformément à une politique de sécurité choisie).

Format de fichier Un format de fichier correspond à un type de fichier particulier qui peut être décodé et interprété par une application donnée ou le système d'exploitation. Un format est généralement associé à une structure interne précise ou à une syntaxe, ainsi qu'à un nom de fichier se terminant par une extension donnée. Par exemple le contenu d'un fichier exécutable Windows

doit (au minimum) commencer par « MZ », et son nom doit se terminer par l'extension « .exe ».

Conteneur Un format conteneur correspond à un fichier susceptible de contenir un ou plusieurs autres fichiers, qu'une application est capable d'extraire. Par exemple une archive ZIP est un format conteneur. Dans l'algorithme générique d'ExeFilter, un répertoire est également un conteneur.

Filtre Un filtre est un module logiciel correspondant à un format de fichier donné, capable :

- d'analyser le nom et le contenu interne d'un fichier afin de confirmer s'il correspond au format attendu,
- de détecter la présence éventuelle de contenu potentiellement dangereux,
- de nettoyer ce contenu le cas échéant,
- ou de prévenir que ce contenu ne peut être nettoyé.

Résultat d'un filtre Après l'action d'un filtre sur un fichier, le résultat peut être l'un des suivants :

- « **accepté** » : le fichier correspond bien au format attendu, et aucun contenu potentiellement dangereux n'a été détecté.
- « **nettoyé** » : le fichier correspond bien au format attendu, au moins un contenu potentiellement dangereux a été détecté, tous ces contenus ont été correctement supprimés ou désactivés.
- « **refusé** » : le fichier correspond bien au format attendu, au moins un contenu potentiellement dangereux a été détecté mais n'a pu être correctement supprimé ou désactivé.
- « **non reconnu** » : le fichier ne correspond pas au format attendu par le filtre.

Algorithme de nettoyage L'algorithme global de nettoyage s'applique sur un conteneur (un répertoire ou une archive). Pour chacun des fichiers, on effectue les actions suivantes :

1. On extrait l'extension du nom du fichier.
2. On sélectionne tous les filtres correspondant à cette extension. Par exemple un fichier « rapport.doc » peut correspondre aux formats MS-Word, RTF ou bien texte ASCII.
3. On applique alors chacun des filtres sélectionnés, qui analyse le contenu du fichier, le nettoie si besoin, puis retourne un des résultats définis plus haut.
4. Les résultats des différents filtres sont fusionnés, en gardant le résultat prioritaire.
5. Si un des filtres correspond à un format conteneur comme ZIP, alors on applique récursivement le même algorithme à ce conteneur.

6. Si le résultat final est « accepté » ou « nettoyé », alors le fichier peut être transmis à destination. Sinon il est placé en quarantaine ou simplement supprimé.
7. Au final, on obtient bien un fichier dont on a vérifié la cohérence du nom avec un format, et dont le contenu a été validé et nettoyé.

Filtrages « liste noire » et « liste blanche » Un filtrage de type « **liste noire** » consiste à accepter tout sauf certains contenus explicitement interdits : « tout ce qui n'est pas interdit est autorisé ».

A l'inverse, l'approche « **liste blanche** » n'autorise que certains contenus bien identifiés et interdit tout le reste : « tout ce qui n'est pas autorisé est interdit ».

Pour ce qui concerne les formats de fichiers, ExeFilter permet d'appliquer une politique de filtrage de type liste blanche, alors que la plupart des antivirus et des logiciels d'analyse de contenu classiques sont plutôt basés sur l'approche liste noire.

3.5 Applications

ExeFilter a été conçu comme un module générique pouvant servir à de nombreuses applications, dont voici certains exemples :

Sas de dépollution de supports amovibles Un sas de dépollution est un poste client permettant l'importation de fichiers vers un système d'information sécurisé à partir de supports amovibles (disquettes, CDs, DVDs, clés USB, ...).

Habituellement ce type de solution repose essentiellement sur 1 ou plusieurs antivirus. ExeFilter permet d'appliquer une politique de sécurité plus stricte afin de mieux protéger le système.

Passerelle de transfert de fichiers ExeFilter peut être employé pour appliquer une politique de sécurité sur une passerelle d'interconnexion, lorsque 2 systèmes d'information ont besoin de s'échanger des fichiers.

Pour cela il est par exemple possible de configurer un serveur FTP ou Samba pour qu'il fournisse un répertoire de dépôt accessible en écriture côté émetteur, et un répertoire destination accessible en lecture de l'autre côté. ExeFilter est alors chargé de nettoyer chaque fichier déposé avant de le recopier dans le répertoire de destination.

Passerelle de filtrage de messagerie De la même façon, ExeFilter peut être intégré dans un serveur SMTP relais sur une passerelle de messagerie, afin d'analyser tous les courriels et leurs pièces jointes.

Passerelle de filtrage Web Pour terminer, ExeFilter peut être intégré dans un proxy HTTP pour sécuriser l'accès au Web.

Au contraire des applications précédentes, le caractère synchrone et dynamique de la consultation Web nécessite des performances très élevées. Ce type d'application n'a pas encore été validé avec la version actuelle d'ExeFilter.

Diode + ExeFilter Pour des passerelles d'interconnexion unidirectionnelles comme celles décrites au chapitre « Diode réseau », il est bien sûr envisageable de combiner la diode et ExeFilter afin de protéger à la fois la confidentialité et l'intégrité du réseau haut, tout en assurant des services de transfert de fichiers, de courriels et de sites web.

4 ...Pourquoi le langage Python ?

Les 2 projets Diode et ExeFilter sont développés avec un temps contraint, en tentant de privilégier l'approche KISS (Keep It Simple, Stupid, cf. [KISS]), pour garantir la clarté du code, l'évolutive et la portabilité pendant les siècles à venir. Un code source simple et clair est toujours plus facile à lire, à comprendre, et donc à sécuriser et à auditer. Cela favorise également le partage au sein d'une équipe de développement. Pour cela, Python est aujourd'hui un des langages les plus adaptés, car il fournit une syntaxe claire et des fonctionnalités de haut niveau. Mais ce sont les experts qui en parlent le mieux :

- « Life is short. You need Python. » (Bruce Eckel, cf. [ECKEL])
- « Python est le chemin le plus rapide entre l'idée et le programme. » (Philippe Biondi, Las Vegas)

Plus sérieusement, Python a permis de gagner beaucoup de temps lors du développement de la diode et d'ExeFilter, puisque sa bibliothèque standard fournit de très nombreuses fonctionnalités de haut niveau, comme par exemple la gestion de fichiers ZIP, de contenus HTML, XML, etc.

Pour finir, ce langage a également permis de développer un code très portable, qui fonctionne aussi bien sous Windows, Linux, BSD ou MacOSX (enfin, presque).

5 Remerciements

Tout d'abord un grand merci à Nicolas Aillery et Jean-François Suret, pour toutes les heures passées à développer la première version de la Diode.

Merci également à Arnaud Kerréneur et Tanguy Vinceleux pour leur aide sur ExeFilter.

6 Conclusion

Les 2 projets Diode et ExeFilter présentés dans cet article apportent deux briques innovantes pour bâtir des interconnexions hautement sécurisées entre un

système d'information sensible et un réseau non maîtrisé comme Internet. De nombreuses applications sont possibles et ne demandent qu'à être développées ou améliorées.

La possibilité de diffuser ces projets en tant que logiciels libres via le site http ://admisource.gouv.fr est en cours d'étude. A suivre...

Références

[DSTO1] Data diodes, M. Stevens & M. Pope, 1995, `http://www.dsto.defence.gov.au/publications/2204/`

[DSTO2] An Implementation of an Optical Data Diode, Malcolm W. Stevens, 1999, `http://www.dsto.defence.gov.au/publications/2110/`

[NPUMP] A Network Pump, Myong H. Kang, Ira S. Moskowitz & Daniel C. Lee, IEEE Transaction on Software Engineering, Vol. 22, No. 5, May 1996, `http://chacs.nrl.navy.mil/publications/CHACS/1996/1996kang-ieee.pdf`

[TENIX] Tenix Interactive Link, `http://www.tenix.com/Main.asp?ID=730`

[FLUTE] RFC 3926, FLUTE - File Delivery over Unidirectional Transport, `http://www.ietf.org/rfc/rfc3926.txt`

[PSYCO] Psyco Python « kind of » JIT compiler, `http://psyco.sourceforge.net/`

[PAR2] outils par et par2, `http://www.par2.net`

[SSTIC03] Formats de fichiers et code malveillant, P. Lagadec, SSTIC03, `http://actes.sstic.org/SSTIC03/Formats_de_fichiers/`

[SSTIC04] Filtrage de messagerie et analyse de contenu, P. Lagadec, SSTIC04, `http://actes.sstic.org/SSTIC04/Filtrage_messagerie/`

[VBPREV] Virus Bulletin prevalence table, `http://www.virusbtn.com/resources/malwareDirectory/prevalence/index`

[KISS] Keep It Simple, Stupid !, `http://en.wikipedia.org/wiki/KISS_principle`

[ECKEL] Why I love Python, Bruce Eckel, `http://www.pythoncriticalmass.com/downloads/pub/eckel/LovePython.zip`