INTRODUCTION

Pourquoi se soucier des canaux cachés ?

Depuis leur création les réseaux d'ordinateurs avaient pour but d'échanger des mails des données ou partager des informations entre chercheurs ou dans des locaux des entreprises les enjeux de la sécurité étaient alors négligeable.

Aujourd'hui avec les nouvelles technologies tous les services et tous les réseaux s'interconnectent, les données de toute personnes reliés à ce vaste réseau sont plus au moins accessibles à n'importe qui ainsi la sécurité des différents réseaux est devenu primordiale, pour assurer la sécurité des informations il a fallu alors restreindre le transit des données de ces réseaux.

Dès qu'on parle de transit d'informations , volontairement ou non cela se fait par un canal donc il est possible d'utiliser le flux autorisé pour faire transiter des données arbitraires dont le trafic est interdit , mettant ainsi en place un canal de communication caché d'où l'intérêt de la stéganographie par rapport à la cryptographie vu qu'elle cherche à dissimuler un message là ou la cryptographie cherche à préserver sa confidentialité .

Ainsi tout administrateur consciencieux sait que les canaux cachés existent, et qu'il ne peut pas grand chose pour les détecter. Néanmoins l'objet de toutes les convoitises en sécurité étant l'information, toute forme de fuite peut s'avérer néfaste.

Nous proposons dans ce rapport dans une première partie une définition des canaux cachés et leur classification, ensuite nous montrons où trouver les canaux cachés puis on expliquera la problématique d'un canal caché sur un réseau et on déterminera les aspects de la mise en place d'un canal caché.

Finalement, nous présenterons une description succincte des techniques pouvant être mises en œuvre pour lutter contre les canaux cachés.

DEFINITION

Un canal est un mécanisme de transfert d'information dans un système. Certains d'entre eux exploitent des mécanismes dont le but original n'est pas le transfert d'information. Ce sont les canaux cachés.
Canal caché ou « covert channel » en anglais est un chemin de communication qui n'a pas été initialement prévu et/ou qui n'est pas autorisé pour transférer de l'information et qui, par conséquent, viole la politique de sécurité mise en place. Signalons qu'un canal caché nécessite donc au moins deux intervenants : celui qui donne les informations et celui qui les reçoit.

Prenons l'exemple d'un utilisateur sur un système qui souhaite employer un logiciel sur ce système. Toutefois, il ne fait pas confiance au logiciel, car celui-ci pourrait redistribuer les informations fournies en paramètre au logiciel (par exemple, un calcul de coûts ou une déclaration de revenus). De même, le concepteur du logiciel ne souhaite pas en révéler le fonctionnement.

Donc un canal caché a toujours le même objectif : enfreindre la politique de sécurité, c'est-à-dire exploiter un moyen de communication non prévu.

Classification

Où se cachent les canaux cachés ?

Partout et il existe de nombreux types de canaux cachés.

Covert storage channel : un processus dispose d'un accès, direct ou indirect, à un espace de stockage en écriture, pendant qu'un autre processus dispose d'un accès, direct ou indirect, à ce même espace de stockage, mais en lecture.

Exemple :

Lorsque plusieurs processus doivent utiliser une même ressource (un fichier), il est courant de poser un « verrou » sur ce fichier. Le canal se fait alors par le biais de la présence ou non de ce verrou. Une autre possibilité est d'agir directement sur le support physique (disque dur, disquette, etc.) : si un *cluster* est occupé, cela code un 1, et sinon un 0.

Covert timing channel : ces canaux consistent pour l'utilisateur à mesurer la vitesse à laquelle s'exécute ses processus afin de déduire ce que font d'autres processus à ce moment alors qu'il ne peut pas les observer directement.

- . **Exemple :**

  Si le premier processus sauvegarde un fichier à une « extrémité » du disque dur, le second processus mesure le temps que met la tête de lecture/écriture pour se déplacer. Le premier processus peut aussi surcharger le CPU pour ralentir certaines réponses à des questions posées par le second processus.

Termination channel : un premier processus lance une tâche, le second récupère l'information en vérifiant que cette tâche est achevée ou non.

***Probabilistic channel* :** un processus modifie la distribution de probabilités d'un événement associé à une ressource, le second processus doit alors estimer cette distribution.

Resource exhaustion channel : ce type de canal s'appuie sur la disponibilité ou non d'une ressource donnée.

- **Exemple :**

  Un premier processus peut remplir tout l'espace disque ou en libérer à sa convenance, le second processus récupère l'information en tentant d'écrire sur le disque. Le bit d'information est transmis en fonction de la réponse à la tentative d'écriture.

Power channel : en supposant qu'un processus soit capable de mesurer la consommation électrique, la communication s'effectue en modulant cette consommation en fonction du bit d'information à transmettre.

Dans cette partie nous avons présenté les différents types de canaux cachés les plus courants. Mais dans ce qui suit les canaux cachés considérés sont encore d'une autre sorte :
ce seraient plutôt des canaux cachés qui exploitent les messages d'un protocole pour transférer des données, c'est-à-dire des canaux légitimes

LES CANAUX RESEAU

Maintenant nous allons voir plus en détails comment toutes ces notions se retrouvent dans un environnement réseau

Le principe fondamental de cette transmission repose sur l'absence de vérification de la valeur intrinsèque des données qui transitent. Les diverses implémentations de systèmes de contrôle d'accès reposent, en effet, sur l'abstraction protocolaire, qui voudrait qu'un transfert de données s'appuyant sur les diverses couches du modèle OSI, ne puisse servir qu'à transporter les données prévues. Notre exposé ne se veut pas exhaustif mais tentera de fournir une nomenclature des canaux cachés aussi complète que possible. Si on s'appuie sur la pile TCP/IP, il devient possible de classer les différents types de canaux cachés en trois grandes catégories :

- Les Canaux cachés de la couche Réseau
- Les Canaux cachés la couche transport
- Les Canaux cachés de la couche application

Maintenant on va voir comment les protocoles peuvent être détournés

1/ Au niveau de la couche réseau :

En consultant le RFC791 on peut remarquer que les paquets IP sont composés d'une en tête commune composée de plusieurs champs. Le champ d'identification du paquet IP est codé sur 16 bits et est employé pour identifier un datagramme IP dans un même flot, c.-à-d. un ensemble de datagrammes IP partageant le même quatre-tuple de source et de destination (IP de source, IP de destination et ports source et de destination). La valeur pour ce champ doit être choisie aléatoirement par la source, mais elle peut également contenir une valeur non-aléatoire sans perturber le mécanisme d'IP. Il est alors possible de détourner ce champ pour cacher 16 bits de données et les envoyer à n'importe quel autre système sur le réseau. Remarquons qu'il a été proposé de cacher les informations transmises dans ce champ de telle manière qu'on garde les propriétés de suite pseudo-aléatoire du champ d'identification.

Le champ option de l'entête du paquet IP est prévu pour les opérations de service requises ou utiles dans quelques situations bien précises mais inutiles pour les communications les plus communes. C'est un champ très intéressant pour pouvoir faire circuler des informations, car il offre un débit spatial très intéressant.

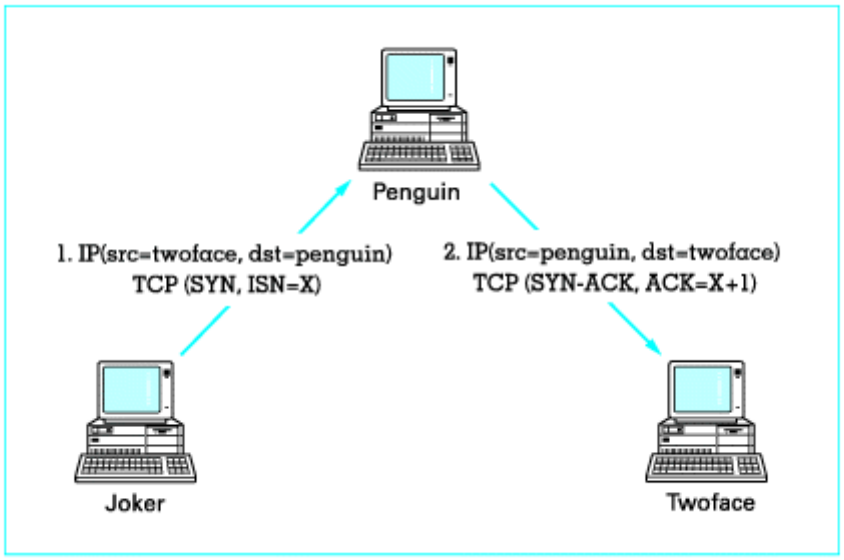
Le champ Flags : est constitué de 3 bits utilisé pour gérer les problèmes de fragmentation est facultatif et sans conséquence sur le fonctionnement du protocole IP.
Et malgré sa taille est petite il reste un moyen d'envoyer quelques bits pas paquet.

2/ Au niveau de la couche transport :

Si le protocole TCP est utilisé le champ contenant le numéro de séquence, codé sur 32 bits, est normalement employé comme numéro d'identification pour réorganiser les paquets au niveau du récepteur lorsque ces derniers arrivent dans le désordre. Il permet et facilite les demandes de retransmission de paquet. Le premier paquet d'une session de TCP (un paquet de type SYN) contient un nombre d'ordre initial aléatoire, ISN. Le destinataire répond typiquement avec un paquet de type SYN/ACK, et en utilisant ISN+1. L'ISN doit être pris de manière aléatoire, cependant, ce champ peut également contenir une valeur non-aléatoire sans perturber le mécanisme de TCP. Il devient donc possible de cacher jusqu'à 32 bits des données dans ce champ et de les envoyer à n'importe quel autre système sur le réseau.

Le champ des options peut lui aussi être employé pour la transmission de données entre l'émetteur et le récepteur.

Utilisation du champ Acquittement avec rebond : Cette fois, la méthode fait participer un troisième intervenant, qui sera chargé de transmettre l'information à la destination.
Le mécanisme est illustré dans la Figure suivante.



Canal caché par rebond

Exemple :

Joker cherche à transmettre un message à Twoface, mais sans se faire remarquer, sans laisser de traces. Pour cela, Joker se sert de Penguin, qui n'a pas besoin d'être au courant : il va transmettre l'information à Twoface malgré lui. La première étape est donc pour Joker de construire un paquet avec le bit SYN au niveau TCP, comme pour ouvrir une connexion TCP. Comme dans le cas précédent, le champ ISN contient la valeur à transmettre moins 1 (nous verrons pourquoi). Ce paquet émis par Joker n'est pas tout à fait normal. En fait, l'adresse source, au niveau IP, n'est pas celle de Joker, mais celle de Twoface.

Lorsque ce paquet arrive à Penguin, il voit qu'il contient le bit SYN et doit donc répondre avec un paquet SYN-ACK dont la valeur d'acquittement est l'ISN plus 1 (d'où le moins 1 précédent). À qui ce paquet va-t-il être envoyé ? La seule information disponible pour Penguin est contenue dans le paquet lui-même : il faut regarder qui a émis ce paquet, c'est-à-dire quelle est son adresse source au niveau IP. Or, Joker a pris soin de remplacer son adresse par celle de Twoface dans le paquet SYN. Donc, quand Penguin examine le paquet, la seule adresse qu'il trouve est celle de... Twoface. Le paquet SYN-ACK est donc renvoyé à Twoface. Pour que cela fonctionne, il faut que le destinataire du message (Twoface ici) soit au courant d'où va provenir le paquet, sinon il passera pour une erreur.

La communication est réalisée entre Joker et Twoface, en se servant de Penguin, qui ne peut être considéré comme complice volontaire. Notons qu'aucune trace directe n'existe de cette relation entre Joker et Twoface.

Cette troisième méthode soulève quelques problèmes. Tout d'abord, elle suppose que la machine source construise des paquets en remplaçant l'adresse source (normalement la sienne) par celle de la machine cible (cette technique s'appelle du *spoofing*) afin que la machine « rebond » réponde à la cible et non à la source.

Lorsque le protocole ICMP les messages echo-request du protocole ICMP peuvent contenir n'importe quoi en guise de données. Selon l'utilisation faite de ce canal, si un superviseur écoute les communications sur le réseau, il pourra remarquer que ces paquets transportent des données, qu'elles soient diffusées en clair ou en chiffré.

En effet, une communication de plusieurs centaines d'échanges echo-request / echo-reply est fortement suspecte car la commande ping sert habituellement à émettre juste quelques paquets pour tester si une machine est toujours active sur le réseau.

2/ Au niveau de la couche application:

Le protocole HTTP (HyperText Transport Protocol) est utilisé pour transférer des informations sur Internet. C'est un protocole extrêmement répandu, ce qui en fait un medium de choix pour la création de canaux cachés. HTTP est basé sur un système à base de question-réponse. Bien que la RFC définissant le protocole contienne 6 différents types de requêtes, seules les requêtes GET et POST sont utilisées en pratique.

Différents champs peuvent être détournés pour créer un canal caché, par exemple les champs HTTP request: {General, Request, Entity}-Header. Ces champs peuvent contenir différents entêtes, comme par exemple « User-Agent: », « Referer: », « Cookie: », on peut ajouter n'importe quel type d'entête pour faire circuler de l'information. Le champ HTTP request Entity-Body n'est normalement présent que dans la requête POST, cependant, la RFC 1945 n'exclue pas la présence de ce champ dans les autres requêtes. Ce champ peut donc servir à transmettre de l'information. Dans les champs de HTTP request, nous retrouverons les mêmes possibilités.

Le protocole DNS, Domain name System, est un protocole de la couche applicative, utilisé pour enregistrer et effectuer des requêtes dans une base de données distribuée des noms de domaine. DNS se base sur un système symétrique de question-réponse.

Une nouvelle fois, les champs de l'entête du protocole vont pouvoir être détournés de leur usage premier pour faire circuler de l'information. Par exemple dans les champs ID, QDCOUNT, ANCOUNT, NSCOUNT, ARCOUNT, QNAME, NAME.

Détection d'un canal caché sur un réseau :

Détection d'un canal caché *via* un message echo-request, et supposons qu'un pirate s'en serve pour transmettre le mot de passe d'un utilisateur qu'il vient de capturer. Le paquet qui passe sur le réseau ressemble alors à cela :

16:03:38.931022 192.168.1.2 > 81.91.65.187:icmp: echo request (DF)0x0000 4500 . .0x0010 . . . . @ . . . @ . . T 0054 0000 4000 4001 e5e8 c0a8 0102E . 515b 41bb 0800 ee34 6e5a 0100 3a9e c33eQ [A . . . . 4 n Z . . : . . >0x0020 S E T S E C R E T C R . . af34 0e00 4352 4554 5345 4352 4554 5345. 4 C R S E C R E T S E E T E0x0030 4352 4554 5345 4352 4554 5345 4352 4554C R E T S E C R E T C R E T0x0040 5345 4352 4554 5345 4352 4554 5345 4352S E C R0x0050 4554 5345E T S E S E

On voit ici le mot de passe (SECRET) répété plusieurs fois dans la charge du paquet, c'est-à-dire du texte clair qui transite sur le réseau. Cependant, quand on examine le paquet émis par la commande ping disponible sur tous les systèmes Unix, on obtient le résultat suivant :

16:01:29.913856 192.168.1.2 > 81.91.65.187: icmp:echo request (DF)0x0000 4500 0054 0000 4000 . .0x0010 515b 41bb 0800 . . . . @ . . . @ . . T 4001 e5e8 c0a8 0102 E . 83d4 5d5a 0100 b99d c33e Q [A . . . . .] Z . . . . . >0x0020 a0f1 0d00 0809 . .0x0030 1415 1617 . . . . 0a0b 0c0d 0e0f 1011 1213 . . " #0x0040 2425 . ! . . . . 1819 1a1b 1c1d 1e1f 2021 2223 . . 2627 2829 2a2b 2c2d 2e2f 3031 3233 . % & 2 0 1 , – . / * + ( ) ′ 30x0050 3435 36374 5 6 7

En fait, la charge n'est pas construite n'importe comment : les huit premiers octets contiennent une indication de temps (b99d c33 a0f1 0d00), le reste des données est ensuite rempli avec la valeur *i* pour l'octet à la position *i* de la charge (à la position 48 de la charge, on trouve la valeur hexadécimale 0x30 – 48 en décimal – soit le caractère ′0′ indiqué à droite dans la traduction en ASCII).

On constate alors que chiffrer les données contenues dans le message n'empêche effectivement pas un observateur de détecter la présence d'un message puisque le contenu habituel du message est modifié pour contenir une suite d'octets aléatoire.

On réalise alors bien la difficulté de la mise en place de tels moyens de détection dans la mesure où ils dépendent fortement des éléments présents sur le réseau, Toutefois, l'empêchement des canaux cachés, reste faisable sur certains systèmes. Des modèles théoriques de sécurité proposent de garantir la confidentialité des données on va citer quelques modèles :

Modèle classique de Bell et LaPadula :

Dans ce qui constitué la première tentative, bell et LaPadula ont proposé un modèle à plusieurs niveaux de sécurité (public, confidentiel, secret, top secret ….) ou les agents de niveau haut n'ont pas le droit d'écrire dans des ressources de niveau plus bas et ou les agents de niveau bas ne peuvent pas lire des ressources de niveau plus haut.

Modèle de non-interférence :

Le principe du modèle est le suivant. Soient A1 et A2 deux applications s'exécutant respectivement aux niveaux L1 et L2. Si L1=L2, alors A1 et A2 peuvent s'échanger des informations. Si L1>L2, alors le flux d'informations de A2 vers A1 est autorisé. En revanche, tout flux de A1 vers A2 est interdit. Enfin si L1 et L2 ne sont pas comparables, aucun flux entre A1 et A2 n'est autorisé. Plusieurs formalisations du principe de Non-interférence ont été proposées. On présente ici celle de Goguen et Meseguer. Soit SI un système d'information. Ce SI permet aux utilisateurs d'exécuter des applications. Soit Seq(SI) l'exécution d'une certaine séquence d'applications dans SI. Chaque application est exécutée avec un certain niveau de sécurité. Si L est un certain niveau de sécurité, on définit Purge(Seq(SI),L), l'exécution obtenue en supprimant de Seq(SI) toutes les applications qui n'ont pas été exécutées avec un niveau inférieur ou égal à L. Enfin, au cours de l'exécution d'une séquence d'applications, un utilisateur s obtient certains résultats (outputs) fournis par le SI. Les résultats obtenus par s au cours de l'exécution d'une séquence Seq(SI) est notée Output(s,Seq(SI)). On dit alors que le système satisfait la propriété de Non-interférence si pour toute séquence Seq et pour tout utilisateur s, on a : Output(s,Seq(SI))=Output(s,Purge(Seq(SI),L)) où L représente le niveau d'habilitation du sujet s. Intuitivement, cette condition exprime que, pour un utilisateur ayant un niveau d'habilitation L, tout se passe comme si seules des applications de niveau inférieur à L avaient été exécutées dans le système.

La non-interférence caractérisée par un système de types :

Une approche assez différente, proposée notamment par Volpano et Smith, puis raffinée plus tard par Boudol et Castellani, consiste à prouver la non-interférence directement par un système de typage adéquat.
L'idée de Volpano et Smith est de typer les variables suivant leur niveau de sécurité, puis de typer les morceaux de programme de telle sorte qu'il soit impossible

– d'une part d'affecter une expression de type haut à une variable de type bas
– d'autre part d'affecter des variables de type bas dans une portion de programme gardée par une expression de type haut.
On interdira par exemple, si l est une variable de type bas et h une variable de type haut, des expressions de la forme :

```
l := 2 ( l + h)
if h = 0 then l := 1 else l := 2
```

Conclusion

A ce jour un grand nombre de recherches dédiés aux canaux cachés sont disponible en tant qu'articles ou en tant qu'outils mais en tant que aspects pratique et application, on n'a pas encore une stratégie efficace pour éliminer le transit des informations avec ses canaux cachés.

Ainsi dans notre projet on a essayé de faire une approche global sur les canaux cachés pour mieux comprendre leur problématique, de plus on a essayer de programmer et d'implémenter l'exemple abordé dans la section canaux réseaux.

Bibliographie :

[1] Aldric Degorre - ENS Cachan, antenne de Bretagne, Caractérisation des canaux cachés en logique temporelle alternante, juin 2005.

[2] Béatrice Bérard, Serge Haddad, Mathieu Sassolas, Canaux cachés temporisés, janvier 2009.

[3] http://www.orbac.org/index.php?page=documentation&lang=fr#NI

[4] RFC, Information Sciences Institute, University of Southern California, septembre 1981.

[5]http://wolverinex02.googlepages.com/lescanauxcach%C3%A9s%28minir%C3%A9seau%29

[6] Frédéric RAYNAL**,** Canaux cachés, octobre 2003.

[7] Raymond, Introduction à la Cryptographie, février 2009.

[8] Timing attack et hyperthreading, Misc 20.