14 sept 2008

Canaux cachés (ou furtifs)

Catégorie : Sécurité Tags : misc

~~Retrouvez cet article dans : **Misc 18**~~

Introduction
Pourquoi se soucier des canaux cachés ? En bon paranoïaque, une réponse pourrait être : « ils sont partout !!! ». Dans le cas d'une crise légère de paranoïa, la solution est alors un emploi systématique de chiffrement (et cryptographie par extension). Néanmoins, cela ne peut satisfaire tous les besoins :

- L'utilisation de chiffrement dissimule le contenu de l'échange entre les entités, mais pas l'existence de cet échange, ce qui s'avère déjà une information intéressante en soi.
- Dans certains pays, des lois sont trop contraignantes sur l'utilisation de cryptographie (limitation des algorithmes ou clés, obligation de déposer les clés aux autorités, etc.), voire en interdisent l'usage.

Prenons le cas de deux prisonniers, Lolo et HB, incarcérés pour détention d'outils de « hack » sans motif légitime. Se sachant menacés, ils ont mis en place un protocole de communication à utiliser lorsque chacun sera dans sa cellule : ils corrompent un gardien avec un Carambar et le charge, en contrepartie, de transmettre une feuille contenant un message chiffré (par exemple, avec un code de César). Si le gardien regarde la feuille, il ne verra que des choses inintelligibles, et suspectera donc que nos deux compères cherchent sans doute à s'évader. En revanche, si la feuille contient une ode en alexandrins, il ne verra pas le vrai message, celui recomposé en mettant bout à bout les premiers mots de chaque vers.
On voit au travers de cet exemple tout l'intérêt de la stéganographie par rapport à la cryptographie. La stéganographie cherche à dissimuler un message, là où la cryptographie cherche à préserver sa confidentialité (caché vs secret).

La face cachée des canaux furtifs

Tout administrateur consciencieux sait que des 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. Dès qu'il y a échange d'information, volontairement ou non, cela transite par un canal. Cet échange demande donc une émission et une réception de l'information en terminologie réseau ou encore une écriture et une lecture en terminologie système.

Un peu de jargon

Tout d'abord, une précision de traduction. Le terme « canal caché » est une traduction maladroite de l'anglais « covert channel ». En effet, covert signifie plus furtif, dérobé que caché. Dans la suite de cet article, nous continuerons à employer le terme consacré en langue française, mais prenez garde à ne pas oublier le sens de l'expression anglaise, plus juste.
Lampson définit en 1973 les canaux cachés comme un canal de communication non prévu pour un quelconque transfert d'information ~~[Lampson73]~~. Nous donnons quelques exemples de tels canaux par la suite.
Cependant la question de la stéganographie réapparaît en 1984, avec le problème des prisonniers (le même que ci-dessus, mais dans une forme bien plus rigoureuse) grâce à Simmons. Il introduit alors la notion de canal subliminal (subliminal channel) ~~[prisoners]~~ : la communication entre les deux prisonniers se faisant de manière ouverte, mais indétectable, il considère que « subliminal » est plus adapté. Il démontre alors comment de tels canaux peuvent être construits dans des schémas de signatures numériques ~~[subliminal]~~.
La stéganographie consiste à dissimuler de l'information dans de l'information, sans éveiller les soupçons, ou encore à injecter de l'information dans un canal furtif afin de dissimuler l'information (voir article dans ce même dossier).
Actuellement, on définit plutôt un canal caché comme un canal de communication non prévu et/ou non autorisé utilisable pour transférer des informations en violation d'une politique de sécurité.
La notion centrale liée aux canaux cachés est celle d'information, et en particulier ses transitions entre différentes entités : on parle alors de flux d'information.
La notion de canal caché est intimement liée à celle de politique de sécurité. Par exemple, lorsqu'on utilise un modèle de sécurité type DAC (Discretionary Access Control), c'est-à-dire qu'une personne gère elle-même ses propres permissions, alors la notion de canal caché n'a pas de sens. En effet, le DAC est « vulnérable » aux malwares puisqu'un tel logiciel peut prendre l'identité d'un utilisateur pour exécuter des actions, mais que le système d'exploitation ne peut distinguer entre l'utilisateur légitime et celui usurpé.
De nombreux autres types de contrôle d'accès existent (MAC, RBAC,...), mais sont mis en œuvre par le système lui-même, l'utilisateur

ne pouvant alors agir sur ses propres permissions (à moins d'avoir les privilèges adéquats). Prenons le cas de 2 politiques de sécurité :

- politique P1 : tous les flux d'information entre les utilisateurs sont interdits.
- politique P2 : une politique de sécurité multi-niveaux qui stipule qu'une information ne peut transiter que d'un niveau donné vers un niveau supérieur.

Avec ces politiques (simplifiées), un canal caché dans P1 peut être totalement légitime dans P2.Plus concrètement, prenons le cas de l'administrateur réseau Lolo qui met en place une politique de filtrage stricte et n'autorise que le trafic HTTP. Malheureusement pour lui, il a parmi ses utilisateurs le petit rigolo HB qui connaît HTTPTunnel et s'en sert pour aller faire de l'IRC avec son copain Jean-Kevin. On voit là encore une violation de la politique de sécurité puisque l'IRC est (censé être) interdit.

Où se cachent les canaux cachés ?

Partout ! Il existe de nombreux types de canaux :
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.
- Par 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 :

- Un processus signale un évènement à un autre processus en modifiant sa propre utilisation d'une ressource système de sorte à modifier le délai de réponse observé par le second processus.
- Par 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 à 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 si cette tâche est achevée ou non.

probabilistic channel :

- Un processus modifie la distribution de probabilités d'un évènement associé à une ressource, le second processus doit alors estimer cette distribution.

ressource exhaustion channel :

- Ce type de canal s'appuie sur la disponibilité ou non d'une ressource donnée. Par 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 est capable de mesurer la consommation électrique, la communication s'effectue en modulant cette consommation en fonction du bit d'information à transmettre.

Le vocable employé ici est orienté système et correspond à celui employé dans ~~[lampson73]~~ en 1973 ! La plupart de ces notions sont bien évidemment transposables à un environnement réseau.
Ce n'est pas parce qu'ils sont cachés que ces canaux n'ont pas de caractéristiques communes avec les canaux de communication usuels :

- La capacité d'un canal de communication est la quantité d'information qu'il peut faire transiter.
- Le bruit intervenant dans un canal de communication correspond aux perturbations engendrées sur l'information lorsqu'elle transite dans le canal.
- Un canal de communication peut fonctionner en mode synchrone (la transmission est quasi-instantanée entre l'émetteur et le récepteur) ou asynchrone (l'émetteur transmet l'information, mais ne peut garantir quand le destinataire la recevra).

La capacité est une mesure importante de la qualité d'un canal. En particulier dans le cas de la sécurité, plus un canal caché permet l'évasion d'information, plus il représente une menace sérieuse. Toutefois, il existe un lien fort et évident entre capacité et furtivité : plus la quantité d'information qui transite illégitimement dans le canal est importante, moins il risque de rester caché longtemps.
La plupart des covert storage channels sont très faiblement bruités dans la mesure où la création d'un objet (ficher, verrou,...) sur un disque peut ne pas réussir si le disque est plein, les permissions ont changé,... mais cela reste peu probable. À l'inverse, les covert timing channels sont souvent fortement bruités, de nombreux facteurs externes pouvant influencer un temps d'exécution ou un calcul.
Un système qui « laisse fuir » ne serait-ce qu'un bit d'information à intervalle régulier est suffisant pour mettre en œuvre un canal de communication. Il suffit de trouver un codage de l'information adapté. Supposons qu'on souhaite disposer d'un alphabet comportant toutes les lettres et les chiffres, soit 36 symboles, un codage simple établit la correspondance A/1, B/2, C/3,... Z/26, 0/27,... 9/36. En codant ces nombres sur 6 bits, on dispose alors de 2^6=64 valeurs possibles, ce qui est largement assez par rapport aux 36 dont on a besoin. L'émetteur pour transmettre la lettre A enverra 000001 (1 en binaire). De son côté, le récepteur sait qu'il devra décoder par bloc de 6 bits. Que les puristes nous pardonnent cette introduction simpliste de la théorie de l'information !!! La chose importante à retenir est qu'il faut bien distinguer ce qui transite dans le canal (une suite de bits) de ce qu'ils représentent (ici, il y a une correspondance directe entre 6 bits et un caractère, mais il pourrait tout aussi bien s'agir d'une image compressée et chiffrée).

Comme nous l'avons déjà laissé entendre, vouloir à tout prix se prémunir des canaux cachés est illusoire : beaucoup trop d'entités interviennent dans un système (au sens large). Les fuites d'information constituent donc un risque à prendre en compte.
À ce titre, les diverses SPA, DPA et autres timing attacks sont des exemples fort instructifs. Lors de la conception des cartes à puces et des programmes qui vont avec, ces attaques n'ont pas été envisagées. Il faut dire : penser à mesurer la température, le temps d'exécution ou la consommation électrique pour reconstruire une clé secrète, il fallait y penser ! Néanmoins, une fois ces fuites identifiées, les solutions pour les boucher ne se sont pas faites attendre.
Mais voyons maintenant plus en détails comment toutes ces notions se retrouvent dans un environnement réseau.

Canaux réseau

Il apparaît comme une évidence que l'une des applications les plus prisées des canaux cachés est la transmission discrète d'information d'un système à un autre, via un réseau partagé et potentiellement surveillé. Certains motifs sont évidents. Il s'agit en général de transférer, à l'insu de tous, des données confidentielles ou de contourner une politique de sécurité interdisant l'usage de telle ou telle application.
Mais en y réfléchissant bien, les canaux cachés offrent la possibilité de fournir des services un peu plus originaux. Ainsi l'exécution d'une séquence d'ouverture d'un port knocker ou le lancement d'ordres d'un maître à ses zombies dans le cadre des dénis de service distribués gagnent énormément à utiliser les canaux cachés. Ces derniers permettent dans un premier temps d'éviter la détection et dans une autre mesure de rendre plus complexe le rejet (intéressant pour les ports knocker) ou le traçage de la source (pour les DDoS).

État de l'art

Le premier article de recherche sur le thème des canaux cachés dans les réseaux date de 1987 ~~[Girling87]~~. Il met en évidence 3 canaux cachés (2 storage et un timing) pour démontrer les possibilités associées à la bande passante disponible sur le réseau local. Bien que ces canaux ne mesurent pas l'impact de leur utilisation sur les performances réelles du réseau, cet article a ouvert la voie à la recherche des canaux cachés dans des environnements type réseau.
Ces travaux ont été poursuivis dans ~~[wolf89]~~ où l'auteur s'est attaché à montrer les possibilités des protocoles courants dans les LAN, type IEEE 802.2, 802.3, 802.4 et 802.5. Il présente comment utiliser des champs non définis de ces protocoles ou le bourrage, pour la mise en place des transmissions. Toutefois, les solutions proposées sont assez faibles car facilement détectables pour quelqu'un surveillant leur utilisation.
Dans ~~[Handel96]~~, les auteurs s'intéressent au modèle OSI et parcourent chaque couche du modèle à la recherche de possibles moyens de communications. Ils en identifient beaucoup, la plupart fondés sur des octets non employés dans les en-têtes. Cependant, on peut adresser deux reproches à cette étude. D'une part, elle reste uniquement théorique puisque s'appuyant sur un modèle théorique décrivant les réseaux. D'autre part, ils ne prennent pas en considération les conséquences de l'utilisation des canaux découverts sur le réseau en terme de performances ou
d' interopérabilité.
Citons également ~~[Rowland96]~~ qui propose des modifications des en-têtes IP et TCP pour construire un canal de communication. Toutefois, là encore, l'auteur ne tient pas compte de l'environnement des paquets (problème de filtrage des paquets, de NAT ou de routage).

Techniques de base

Le premier point essentiel dans la création d'un canal réseau est de s'assurer que les paquets destinés à transporter l'information sont parfaitement normaux : d'une part, ils doivent se conformer à la lettre aux RFC et, d'autre part, ils correspondent à des protocoles ou applications légitimes sur le réseau. Faute de répondre à ces deux critères, ils risquent de générer une alerte et par conséquent d'éveiller la suspicion.
Il apparaît ainsi naturel que le premier protocole à avoir été utilisé à ces fins soit ICMP. En effet, ce dernier présente deux avantages majeurs :

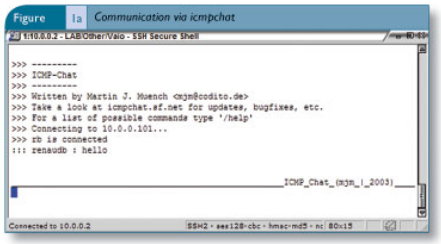
- Il a longtemps été considéré comme anodin et inoffensif (et l'est toujours dans bien des cas).
- Il définit un champ de données allant de 84 octets

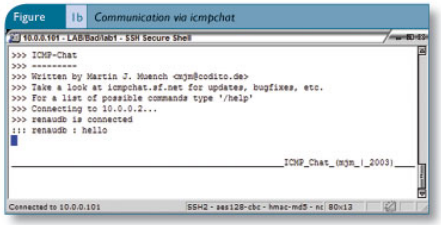
(ICMP Redirect ou Destination Unreachable par exemple) à MTU-28 octets (ICMP echo request/reply sans fragmentation).
Une implémentation que l'on pourrait qualifier de triviale est fournie par ~~icmpchat [icmpchat]~~. Ce petit programme est plus ou moins l'équivalent d'un talk utilisant le type et le code de message au choix de l'utilisateur et chiffrant la communication. Les figures 1a et 1b montrent l'interface de chaque côté du canal. La figure 2 donne l'analyse du paquet de transport de la communication : il s'agit bien d'un paquet ICMP légitime dont les données ne sont pas explicites.

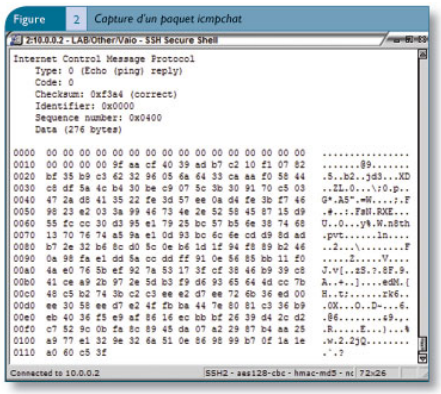
Protocoles réseau

L'utilisation du champ de données d'un protocole n'est cependant pas la méthode la plus subtile pour transférer des informations. Si ces dernières ont peu de chance d'être comprises compte tenu du chiffrement, l'afflux de paquets ICMP – pour reprendre l'exemple précédent – contenant des données peut apparaître en soit comme une anomalie. Mais plus généralement, la capacité de ces canaux est extrêmement réduite (ce qui peut néanmoins largement suffire, par exemple pour transférer une clé de chiffrement de quelques octets). Dans ces conditions, il s'avère intéressant de trouver d'autres champs autorisant la transmission d'informations.
Les conditions d'utilisation de tels champs sont les suivantes :

- Ils doivent être « manipulables » à volonté.
- L'objectif du canal est de transmettre relativement rapidement et le plus discrètement possible des volumes de données importants. Pour cela, il est préférable que le spectre des valeurs que peuvent prendre les champs soit le plus large possible.







- Ils ne doivent pas être modifiés systématiquement de manière aléatoire au cours du transfert. En effet, en cas de modification « accidentelle », il s'agit du bruit, dont l'impact dépend des mécanismes de correction mis en place. Prenons cependant le cas de la translation d'adresse dynamique. Lorsque le paquet de l'émetteur traverse le NAT, le port source est systématiquement modifié, et la valeur qui va lui être attribuée ne peut être prédéterminée. Dans ce cas l'utilisation du port source comme champ de transport de l'information est évidemment impossible.

Il faut bien voir que pratiquement tous les champs d'un paquet sur un réseau peuvent servir, de manière plus ou moins adaptée, à l'élaboration d'un canal caché. Néanmoins, en tenant compte des aspects pratiques (capacité, furtivité, etc.), on se rend compte que le choix est quand même limité.

L'en-tête IP

Nous aborderons rapidement les cas des options, de l'identifiant (IPID) et de l'adresse source.
L'exploitation des options nécessite leur support par les piles réseau source et destination ainsi que la garantie qu'aucun équipement de filtrage ne les modifie ou ne les supprime, ainsi que le ferait par défaut un firewall CheckPoint. Ce champ n'est donc pas idéal, loin s'en faut.
L'adresse source en revanche apparaît déjà comme plus stable dans la mesure où elle ne risque d'être modifiée qu'en cas de traversée d'un NAT, qu'il soit statique ou dynamique. En revanche, une communication bidirectionnelle ne sera rendue possible que si chaque extrémité du canal est codée en dur dans le programme, l'information concernant la source étant évidemment perdue en cours de transfert.
Il n'en reste donc qu'un, l'IPID, largement utilisé [ipid][stegtunnel] mais qui présente un inconvénient de taille : il ne permet que le transfert de deux octets par paquets. Ainsi, s'il peut être exploité avec succès pour le lancement d'une commande ou l'exécution d'une séquence de port knocking, il s'avère tout à fait inadapté au transfert de volumes importants de données.

En-tête UDP

Sur les quatre champs d'en-tête, trois sont exploitables : le port source, la longueur des données et le checksum. Le port source est une option intéressante dans la mesure où il est uniquement sensible au NAT dynamique. Les autres champs peuvent également être exploités si aucun équipement de détection n'effectue de vérification au niveau 4, ce qui est généralement le cas. En utilisant l'ensemble des champs, il est donc possible de transférer 12 octets par paquet, sachant qu'une fois encore aucune garantie d'intégrité de la communication n'est fournie par le protocole.
Cet inconvénient est également son principal avantage dans la mesure où les moteurs de filtrage de type stateful auront beaucoup plus de mal à détecter une anomalie que lors de la violation d'un handshake TCP.

En-tête TCP

Beaucoup plus riche, cet en-tête présente de nombreux champs exploitables pour la dissimulation des communications : les numéros de séquence, la fenêtre, le port source, le checksum, les flags et les options, plus particulièrement le time stamp. La charge théorique de chaque paquet s'élève par conséquent à 38 octets par paquets.
Néanmoins, les moteurs de détection d'anomalies de type stateful ou de compatibilité avec les RFC réduisent considérablement cette charge. En effet, une session TCP doit commencer par un SYN (accessoirement avec le flag Push positionné) et le numéro de séquence d'acknowledgement être 0. La charge initiale se trouve alors réduite à 32 octets et 2 bits. En outre l'évolution des numéros de séquence doit suivre un certain nombre de règles, réduisant encore une fois la charge et le spectre possible des valeurs transmises. Enfin, les mécanismes de lutte contre les SYNFloods tels que les SYNcookies interdisent complètement l'utilisation du numéro de séquence d'acknowledgement à des fins de transmission d'information.

Canaux applicatifs

Au niveau applicatif les canaux cachés vont permettre d'utiliser un flux parfaitement légitime tant en termes de respect de la politique de sécurité qu'en termes d'application stricte des normes de communication. En outre, ce type de flux paraîtra moins suspect dans la mesure où un trafic HTTP important semble toujours plus naturel qu'un volume conséquent de paquets ICMP…
Compte tenu des restrictions liées aux flux autorisés généralement en sortie, les principaux trafics applicatifs exploités sont le Web, le mail et le DNS. Associés aux différents mécanismes de relais tels que des chaînes de proxy pour le Web ou des remailers anonymes pour le mail, ces canaux garantissent l'intégrité (au sens TCP du terme) des transferts, un bon débit de données ainsi qu'une résistance certaine au traçage de leur source. En outre, compte tenu de la richesse des protocoles applicatifs, les possibilités sont quasi-infinies et la détection pratiquement impossible. Toutefois, un flux HTTP qui dure plus de quelques minutes n'est pas tout à fait normal, tout comme une augmentation subite du trafic DNS.
Un exemple de ce type de canal, fondé sur la résolution de noms, est détaillé plus avant dans ce numéro.

Utilisation des rebonds

Outre la taille réduite du canal de communication, les techniques précédemment décrites présentent également l'inconvénient de permettre une remontée triviale à la source du canal. Il devient alors possible de bloquer définitivement la source au niveau d'un équipement de filtrage et de rendre le canal définitivement inutilisable. Les rebonds profitent d'une tierce partie qui transmet, à son insu, l'information entre la source et la destination. En voici quelques exemples.

Rebond par SYN/ACK

Conformément à la RFC, le numéro de séquence transmis par un client dans un paquet SYN (appelé ISN, Initial Sequence Number) à destination d'un port ouvert est incrémenté de 1 par le serveur et retransmis au client à travers le numéro de séquence d'acknowledgement. Sur cette base, il est tout à fait possible de transmettre une information en utilisant un serveur quelconque et en spoofant l'adresse IP source afin de la faire pointer sur l'autre extrémité du canal.
Exemple : La machine A (10.0.0.1) veut communiquer avec B (192.168.0.1) via le serveur web C (www.prendsmoipouruncon.com) pour lui transmettre le message ~~service ssh start\n~~, soit 18 octets, encodé en hexa de la manière suivante : ~~\73\65\72\76\69\63\65\20 \73\73\68\20\73\74\61\72\74\0A~~ et divisé en cinq blocs de 4 octets (taille du numéro de séquence) :

- séquence 1 : ~~0x73657276~~ ;
- séquence 2 : ~~0x69636520~~ ;
- séquence 3 : ~~0x73736820~~ ;
- séquence 4 : ~~0x73746172~~ ;
- séquence 5 : ~~0x740A0000~~ - deux octets nuls ont été utilisés pour le padding.

Afin d'établir le canal, il suffit d'envoyer 5 SYN à destination du serveur C sur le port 80, avec comme source l'adresse IP de B (192.168.0.1) et avec les numéros de séquence précédemment calculés. C renverra à la source (B) un SYN/ACK avec comme numéros de séquence d'acknowledgement respectifs pour chacun des 5 paquets ~~0x73657277~~, ~~0x69636521~~, ~~0x73736821~~, ~~0x73746173~~, ~~0x740A0001~~, soit les 5 numéros de séquence incrémentés de 1. Le réassemblage est alors trivial.
Cependant, des paquets RST sont renvoyés par la destination à la machine ayant servi de rebond. Cela implique qu'on doit prendre quelques précautions dans le choix de cette machine. De plus, le fait de voir sur un réseau des paires de paquets SYN/ACK puis RST n'est pas du tout normal et constitue donc une bonne signature pour la détection d'un tel canal.

Rebond par erreurs ICMP

Dans le même ordre d'idée, il est possible de rebondir en s'appuyant sur les messages d'erreur ICMP. En effet, pour les messages de type ICMP destination/port unreachable, ICMP time exceeded et ICMP redirect, la RFC précise que le champ de données ICMP doit contenir l'en-tête IP du paquet ayant généré l'erreur ainsi que les 64 premiers octets de données.
La première application est triviale pour les stations A et B souhaitant transférer des informations via C :

- A envoie un paquet provoquant volontairement une erreur. La source du paquet est B. Les données à transmettre sont dans les 64 premiers octets des données IP.
- Par un moyen ou par un autre (voir plus loin) le système C identifie une erreur. Il envoie le paquet ICMP approprié à la source, soit B.
- B réceptionne le paquet et récupère les données.

La génération de telles erreurs est relativement simple. Par exemple, un paquet ~~ICMP port unreachable~~ est envoyé lors de l'émission d'un paquet UDP contenant les données à transmettre sur un port fermé. Dans ce cas, le serveur de rebond C est un serveur quelconque sur Internet auquel notre paquet sera envoyé sur un port élevé (donc probablement fermé). Dans ce cas, notre paquet doit être correct au niveau UDP, ce qui fait perdre 16 octets supplémentaires. Il ne reste donc que 64 - 16 = 48 octets disponibles.

Beaucoup plus intéressant, envoyer un paquet avec un TTL particulièrement faible présente de nombreux avantages. Premièrement, l'erreur est générée dès l'analyse de l'en-tête IP. Il est donc inutile de structurer les données IP conformément à tel ou tel protocole de niveau 4 et la charge totale par paquet est donc bien de 64 octets. Le deuxième avantage de cette technique est que le serveur de rebond C n'a pas à être identifié. En effet, le message ICMP d'erreur sera généré par le routeur voyant le paquet émis par A (avec comme source B) avec un TTL de 0. Il transmettra par conséquent à B (source officielle du paquet) le message d'erreur avec les données. Le serveur de rebond devient donc un routeur quelconque sur Internet. Enfin, et compte tenu du fait que le serveur de rebond n'est plus déterminé ni déterminant, la destination même du paquet n'a plus d'importance. Cela permet de réduire encore une fois les possibilités de « traçage » des canaux dans la mesure où même le serveur de rebond n'est plus fixé.

Application au portknocker

Prenons l'exemple d'un administrateur avisé que nous appellerons Lolo. Ce dernier utilise SSH pour se connecter à ses systèmes. Il est néanmoins conscient que la version qu'il utilise comporte sûrement des failles non publiées. En outre, il sait pertinemment que, de toute façon, il n'aura jamais le temps de mettre à jour tous ses serveurs dans un temps raisonnable une fois l'exploit dans la nature, et qu'il sera alors victime du premier script kiddy venu, que nous appellerons Fred. Par conséquent, il a choisi de mettre en place un portknocker qui, à l'issue d'une certaine combinaison de paquets, permet le lancement du service en question. Soucieux d'éviter tout rejeu de ces sessions par un méchant w4rl0rDz que nous appellerons HB, il souhaite également mettre en place un canal caché avec rebond multiplexé.
Parmi la pléthore de portknockers, il choisit apk ~~[apk]~~. Au niveau du serveur, le fichier de configuration (~~apks.conf~~) est le suivant :

```
#TYPE   ID      TTL     SOURCE  SPORT   DPORT   SEQ     SEQACK  URG     ACK     PSH     RST     SYN     FIN     WIN     DATA    ACTION  KEY
E               1       *       *       80       *      *       12345   0       1       0       0       1       0       *       *       NEXT(2) NULL
F               2       *       *       80      *       *       67890   0       1       0       0       1       0       *       *       NEXT(3) NULL
F               3       *       *       80      *       *       13579   0       1       0       0       1       0       *       *       CMD("service ssh start")        I
```

Ainsi le service SSH sera lancé si le portknocker reçoit 3 SYN/ACK consécutifs provenant de serveurs web (SPORT = 80) et dont les numéros de séquence d'ackowledgement sont respectivement 12345, 67890, 13579. Afin d'automatiser ces opérations, le fichier de configuration du client sera :

```
        #TYPE   ID      TTL     SOURCE   SPORT  DPORT   SEQ     SEQ_ACK  URG     ACK     PSH     RST     SYN     FIN     WIN     DATA    ACTION  KEY
        E       1       *        *               *      80       12344  0       0       0       0       0       1       0       *       *       NEXT(2)         NULL
        F       2        *       *       *      80       67889  0        0       0       0       0       1       0       *       *         NEXT(3)               NULL
        F       3        *       *       *      80       13578  0        0       0       0       0       1       0        *      *       CMD(«service ssh start»)        I
```

De cette manière, chaque étape sera programmée pour envoyer en séquence trois paquets SYN sur le port 80 de serveurs (spécifiés manuellement à chaque étape) et avec des numéros de séquence correspondant à ceux du serveur, moins 1.

Rebonds applicatifs

Le même principe peut être appliqué au niveau applicatif ou presque. La limitation vient du fait que le mode d'établissement d'une session TCP présente une difficulté d'exploitation telle qu'elle en devient rédhibitoire. UDP en revanche reste un socle privilégié que nous n'allons pas nous priver d'exploiter au travers d'un exemple simple et ludique : SIP. Sans rentrer dans les détails du protocole, rappelons rapidement les principaux champs de l'en-tête SIP.

```
1 INVITE sip:bob@sipserver.com SIP/2.0
2 Via: SIP/2.0/UDP www.prendsmoipouruncon.com;branch=z9hG4bK776asdhds
3 Max-Forwards: 70
4 To: Bob <sip:bob@sipserver.com>
5 From: Renaud <sip:renaud.bidou@prendsmoipouruncon.com>;tag=1928301774
6 Call-ID: a84b4c76e66710@www.prendsmoipouruncon.com
7 CSeq: 314159 INVITE
8 Contact: <sip:admin@prendsmoipouruncon.com>
9 Content-Type: application/sdp
10 Content-Length: 142
```

L'en-tête précédent est une demande de connexion. Pour simplifier, nous pouvons la comparer à un SYN dans le cas d'une ouverture de session TCP. À la ligne 1, l'en-tête contient la commande ~~INVITE~~ caractéristique d'une demande de connexion ainsi que l'identifiant de l'utilisateur destinataire. La ligne 2 fournit l'adresse www.prendsmoipouruncon.com à laquelle la réponse à l'invitation doit être envoyée. Enfin ligne 7, un numéro de séquence de commande sur 4 octets qui sera repris dans la réponse à la requête.
SIP ne prenant absolument pas en compte les informations IP à cause des différents mécanismes de proxy, le canal et la méthode de rebond apparaissent clairement. Le numéro de séquence de commande SIP va contenir les données, le serveur de rebond est un serveur SIP et la destination de la réponse spoofée d'un point de vue applicatif grâce à la modification du champ Via à la ligne 2.

Rebonds multiplexés

Les techniques de rebond décrites ci-dessus peuvent être encore améliorées grâce à une technique de multiplexage simple mais efficace. L'objectif n'est plus d'utiliser 1 mais n serveurs de rebonds.
Reprenons notre exemple avec les 5 numéros de séquence TCP. Au lieu de rebondir sur 1 serveur web, rien n'empêche d'exploiter 5 serveurs web différents. Cela impose de respecter un certain délai entre chaque paquet mais encore une fois les opérations de détection et de traçage s'avèreront bien plus compliquées.

Toutefois, cela nécessite la mise en place d'un protocole de communication complexe afin que la destination soit capable de :

- distinguer un paquet normal d'un paquet porteur d'information ;
- réordonner les paquets porteurs d'information afin de reconstituer dans le bon ordre l'information complète.

Conclusion

Au travers des exemples précédents, on distingue deux « endroits » où mettre en place un canal caché :

Les données utilisées:
Souvent, elles doivent être mises à 0 dans les paquets et leur taille est relativement réduite ;

des champs spécifiques dans les en-têtes des protocoles :
Il faut alors prendre en considération les impacts sur l'environnement afin de rester le plus furtif possible, ce qui demande donc une connaissance précise du protocole manipulée.

La question qui se pose ensuite est celle de la qualité d'un canal : selon quels critères peut-on « qualifier » un canal caché ? Cette démarche sert à déterminer les caractéristiques essentielles du canal caché. On peut alors en déduire les précautions à mettre en place (chiffrer ou non, code correcteur ou non, etc.) pour assurer la discrétion de la communication.
Les cinq caractéristiques principales sont :

La capacité :
Il s'agit du taux de bits d'information transportés. Par exemple, la taille minimale d'un paquet Ethernet est de 64 octets, contre 2 pour le champ IPID, soit une capacité de 2/64=0.03125 bits.
la détectabilité :
Dans quelle mesure ce canal est-il facile à détecter ? L'utilisation du TCP ISN change selon les systèmes d'exploitation. Ce numéro est aléatoire pour les BSD et convient donc parfaitement à des données chiffrées (puisqu'elles semblent aléatoires). Ce n'est en revanche pas le cas avec Windows 98 où l'ISN est incrémenté de 1 entre chaque nouvelle connexion.

la féquence :
Elle correspond au nombre de fois qu'un canal peut être utilisé par session. Dans le cas d'une connexion TCP et de l'emploi de l'ISN, un seul paquet contient l'information. Au contraire, avec l'IPID, chaque paquet emporte de l'information.

les conditions d'emploi :
Il s'agit des choses impératives pour que le canal fonctionne, comme une session TCP pour utiliser des SYN/ACK avec rebonds.

les précautions :
Quelles sont les précautions à prendre lors de l'utilisation de ce canal et quels aspects sont à considérer pour qu'il fonctionne correctement ? Dans le cas du TCP ISN, un proxy TCP ou la réécriture des en-têtes comme le permet le pare-feu ~~pf~~.

Les canaux cachés fonctionnent souvent grâce au bricolage et au bon sens. Les détecter est particulièrement ardu, et cela est d'autant plus vrai que peu de données sont transportées. En revanche, un flux HTTP de plusieurs heures, un volume accru de trafic ICMP ou DNS s'avère souvent symptomatique de la présence d'un canal caché.

Références

- [apk] Advanced Port Knocker : http://www.iv2-technologies.com/~rbidou/apk-1.0.tar.gz.
- [Girling87] C. G. Girling, « Covert channels in LAN's », IEEE Transactions on Software Engineering, 1987.
- [Handel96] T. G. Handel et M. T. Sandford, « Hiding Data in the {OSI} Network Model », Workshop on Information Hiding, 1996.
- [icmpchat] icmpchat : http://icmpchat.sourceforge.net/.
- [ipid] passivecc_ipid : http://invisiblethings.org/tools/passivecc_ipid.c.
- [Lampson73] B. W. Lampson, « A Note on the Confinement Problem », Communication of the ACM, 1973.
- [Pinchuk] Evgeny Pinchuk, « Covert Channels in Networking » : http://www.cs.tau.ac.il/tausec/lectures /Network_Covert_Channels.pps.
- [prisoners] « The Prisoners' Problem and the Subliminal Channel », in G.J. Simmons, Proceedings of CRYPTO '88, Plenum Press (1984), pp. 51-67.
- [Rowland96] C. H. Rowlan, « Covert channels in the TCP/IP protocol suite » : http://www.firstmonday.dk/issues/issue2_5/rowland/.
- [subliminal] « The Subliminal Channel and Digital Signatures », in G.J. Simmons, Advances in Cryptology, EUROCRYPT '84, Springer LNCS v 209.
- [stegtunnel] stegtunnel : http://www.synacklabs.net/projects/stegtunnel/.
- [wolf89] M. Wolf, « Covert channels in LAN protocols », Workshop on LAN security (LANSEC'89), 1989.

~~Retrouvez cet article dans : **Misc 18**~~

Posté par (La rédaction) | Signature : Renaud Bidou | Article paru dans misc

Laissez une réponse

Vous devez avoir ouvert une session pour écrire un commentaire.

« Précédent Aller au contenu »

Identifiez-vous
Inscription
S'abonner à UNIX Garden

Articles de 1ère page

- Dos par complexité
- Petite introduction à l'électronique à l'usage des sysadmins et codeurs
- Sortez du software !

- Les dénis de service
- La mort annoncée du WEP
- nikto Tests de serveurs HTTP
- duplicity Sauvegarde chiffrée
- knl
- ffmpeg
- tor

Actuellement en kiosque : 

Il y a actuellement

771 articles/billets en ligne.

Recherche

Catégories

- Administration réseau
- Administration système
- Agenda-Interview
- Audio-vidéo
- Bureautique
- Comprendre
- Distribution
- Embarqué
- Environnement de bureau
- Graphisme
- Jeux
- Matériel
- News
- Programmation
- Réfléchir
- Sécurité
- Utilitaires
- Web

Archives

- septembre 2008
- août 2008
- juillet 2008
- juin 2008
- mai 2008
- avril 2008
- mars 2008
- février 2008
- janvier 2008
- décembre 2007
- novembre 2007
- février 2007

- **GNU/Linux Magazine**

- GLMF, partenaire de l'évènement "Paris, capitale du Libre"
  - GNU/Linux Magazine 108 - Septembre 2008 - Chez votre marchand de journaux
  - Edito : GNU/Linux Magazine 108
  - GNU/Linux Magazine HS 38 - Septembre/Octobre 2008 - Chez votre marchand de journaux
  - Edito : GNU/Linux Magazine HS 38

- **GNU/Linux Pratique**

- Linux Pratique soutient la journée mondiale contre les brevets logiciels
  - Linux Pratique, partenaire de l'évènement "Paris, capitale du Libre"
  - Linux Pratique N°49 -Septembre/Octobre 2008 - Chez votre marchand de journaux
  - Edito : Linux Pratique N°49
  - À télécharger : Les fichiers du Cahier Web de Linux Pratique n°49

- **MISC Magazine**

- Misc 39 : Fuzzing - Injectez des données et trouvez les failles cachées - Septembre/Octobre 2008 - Chez votre marchand de journaux
  - Edito : Misc 39
  - MISC 39 - Communiqué de presse
  - Salon Infosecurity & Storage expo - 19 et 20 novembre 2008.
  - Misc 38 : Codes Malicieux, quoi de neuf ? - Juillet/Août 2008 - Chez votre marchand de journaux