La lutte contre les dénis de service réseau

Renaud Bidou

Radware
renaudb@radware.com

Résumé Les dénis de service ne sont pas nouveaux. Les principales techniques utilisées ont parfois plus de dix ans. Néanmoins l'accroissement sensible des attaques de ce type, généralement à des fins d'extorsion, prouve d'une part qu'elles restent d'actualité et d'autre part que les solutions proposées aujourd'hui ne permettent pas de se protéger efficacement. Nous verrons dans un premier temps les aspects techniques de telles attaques en détaillant leur fonctionnement et leur impact sur les composants du système d'information. Nous essaierons ensuite de trouver des solutions techniquement pertinentes, industriellement fiables et applicables dans le cadre d'un réseau d'entreprise.

1 Historique

Si les SYNFloods sont nés avec TCP d'autres attaques ont vu le jour petit à petit, regroupées en deux grandes familles : les anomalies réseau et l'épuisement de ressources.

La première apparition des attaques par anomalies remonte à 1995 avec le fameux « ping de la mort », déstabilisant le stack grâce à la fragmentation ICMP. Quelques années plus tard les « bo(i)nk » et autres « teardrop » suivaient cette voie en jouant cette fois avec la fragmentation UDP. Depuis l'exploitation systématique des exceptions et autres cas spécifiques ou non définis du stack fournit encore de nombreuses solutions de DoS.

L'épuisement des ressources a vue le jour avec les SYNFloods, et bien que ce type d'attaque reste particulièrement efficace d'autres approches ont été utilisées essentiellement dans le but d'augmenter l'efficacité de l'attaque. Ainsi les années 90 ont été le théâtre des bombes logiques du type « echo-chargen » ou encore des premières techniques « d'effet de levier » telles que celles utilisées par « smurf » dès 1998.

Depuis peu de techniques ont changé. Cependant deux éléments essentiels sont à l'origine d'évolutions notables. Le premier est l'accroissement des performances des systèmes et de leurs moyens de communication. Quand un 486 générera difficilement plus d'un millier de SYN par seconde, un PIV est à même d'en générer plus de 200.000. Une évolution du même ordre est à prendre en compte en termes de vitesse d'accès et de capacité de transport d'Internet.

Le second facteur est la dépendance accrue des entreprise à leur outils de communication informatiques tels que les ERP, l'e-mail ou encore la voix sur IP, quand l'activité de l'entreprise ne dépend pas entièrement de ces moyens de

communication. Cette exposition attire naturellement depuis quelques années les professionnels de l'extorsion tant les sommes en jeux peuvent être considérables. Ainsi certains sites de paris en ligne peuvent générer plusieurs dizaines de millions d'euros de chiffre d'affaire pendant la demi-heure qui précède un match important de coupe d'Europe... La sanction est immédiate.

2 Techniques de DoS

2.1 Épuisement de ressources par anomalies

La voie a été ouverte par le « ping de a mort » qui consiste à envoyer un paquet ICMP de plus de 65535 octets que les stacks IP de l'époque ne savaient pas gérer proprement. Ont suivi les erreurs de fragmentation UDP, ou encore les paquets TCP contenant des « flags » illégaux ou incompatibles. Les effets de telles attaques étaient variés. Certains stacks se contentaient de « freezer » les connexions pendant toute la durée de l'attaque quand d'autres provoquaient un irrémédiable « reboot ».

Les stacks actuels résistent à ce type d'attaques. Néanmoins les temps de traitement de tels paquets restent plus longs que ceux nécessaires pour les paquets légitimes. Ainsi il devient trivial de générer une consommation excessive de CPU par la simple émission de plusieurs centaines de milliers d'anomalies par secondes, ce qu'un outil tels que hping3 permet en une unique ligne de commande...

```
[root@localhost root]#
hping3 -SARFU -L 0 -M 0 -p 80 www.cible.com --flood
```

Concrètement un PIV avec 512 Mo de RAM sous Windows XP supportant apache 2.0.42 atteindra 100% de CPU lors de la transmission d'une centaine de milliers de paquets par seconde avec les flags SYN/ACK/FIN/RST et URG positionnés. Nous verrons plus loin que rajouter le flag PUSH peut avoir un certain intérêt.

L'augmentation du potentiel d'attaque, liée à la démocratisation de l'accès haut débit et de la puissance des ordinateurs personnels a mis en évidence la faiblesse des patchs développés il y a plusieurs années. Ainsi « land », une attaque qui date de 1997, a trouvé une nouvelle jeunesse au point de voir sa version parallélisée « imland » publiée début mars 2005 sur bugtraq.

Cette augmentation de la puissance potentielle permet à quasiment toutes les anomalies d'être à l'origine d'un déni de service, pourvu qu'elles soient générées à un rythme suffisamment important. Ainsi l'usage des champs « réservés » de l'en-tête TCP, le positionnement d'un numéro de séquence d'« acknowledgement » dans un paquet SYN ou encore des paquets dont l'en-tête de couche 4 (TCP/UDP) est tronqué en dépit de checksums corrects sont autant d'anomalies dont les effets peuvent être dramatiques.

2.2 Épuisement de ressources par concept

Les SYNFloods existent toujours et semblent avoir encore de beaux jours devant eux. Ils représentent à ce jour le plus beau cas de Dénis de Service par concept, TCP intégrant nativement la technologie « DoS Inside ». S'il est inutile (et de toute façon interdit, ce afin de protéger nos concitoyens et de maintenir les responsables sécurité de nos grandes entreprises dans une rassurante ignorance) s'il est inutile donc de décrire le mode de fonctionnement de cette attaque, rappelons néanmoins ses principaux points forts.

Tout d'abord un SYN ne fait que 64 octets, ce qui réduit considérablement les besoins en termes de bande passante.

Ensuite, et surtout le SYNFlood n'impose pas à la machine attaquante de maintenir de session. Les conséquences indirectes sont considérables dans la mesure où l'adresse IP source peut être « spoofée ». De cette manière les SYN_ACK émis par la cible seront renvoyés vers d'autres systèmes, et ne viendront pas consommer inutilement la bande passante de l'attaquant.

En y réfléchissant, il est également possible d'augmenter l'impact de l'attaque pour le même prix grâce à la réflexion. Cette technique s'appuie sur le fait qu'un SYN_ACK reçu par un système sur un port fermé ou n'ayant aucun lien avec une connexion existante sur un port ouvert provoque l'émission d'un RST. Dans le cas d'un SYN dont l'adresse source est spoofée mais correspond néanmoins à une adresse existante, le SYN_ACK générera en retour un RST qui viendra s'ajouter à la charge en cours.

Un autre type d'attaque basée sur le concept est l'usage du flag PUSH. En effet lors de la réception d'un paquet, le stack TCP garde ce dernier dans un buffer. Lorsque le buffer est rempli, les données sont « passées » au système. Cette technique permet de réduire l'overhead lié à la transmission de données entre le stack et le système. Néanmoins, si le flag PUSH est positionné le contenu du buffer est immédiatement transmis au système, ce qui génère une charge supplémentaire qui peut s'avérer rapidement considérable et donc fatale pour le système.

Cette attaque est particulièrement efficace sur les services qui maintiennent la connexion entre le client et le serveur, tels que SMTP par exemple. De cette manière la session TCP peut être établie normalement puis des ACK+PUSH lancés à l'intérieur de cette session. Si le travail est fait proprement l'attaque sera particulièrement difficile à détecter dans la mesure où :

1. elle s'appuie sur un trafic autorisé par les systèmes de filtrage ;
2. elle est parfaitement légitime d'un point de vue stateful TCP.

2.3 Épuisement de ressources applicatives

Le réseau n'est pas nécessairement le seul vecteur de génération des dénis de service. Ces derniers peuvent être également générés au niveau applicatif, ce à plusieurs titres.

En premier lieu, il est possible de s'appuyer sur les limitations internes des applications. Ainsi la plupart des serveurs web ont une limite paramétrable en

termes de sessions HTTP concurrentes pouvant être traitées. Le principe du déni de service à ce niveau est trivial. Il suffit d'établir un grand nombre de sessions sur le port 80 du système cible sans jamais les fermer. Les capacités de génération de telles sessions sont en générale suffisamment importantes pour assurer une saturation de la cible avant le « nettoyage » par timeout. Un outil tel que webdevil permet d'effectuer une telle opération à la ligne de commande :

```
[root@localhost root]#webdevil www.target.com 80 10000
```

A un autre niveau les fonctionnements internes d'une application recèlent de nombreux goulets d'étranglements. Prenons cette fois l'exemple des serveurs DNS. Lorsqu'un client émet une requête imposant la réponse d'un serveur faisant autorité à destination d'un serveur qui n'est pas SOA du domaine concerné, ce dernier :

1. recherche le serveur faisant autorité sur le domaine,
2. émet une requête à destination de ce dernier,
3. se met en attente de la réponse,
4. transmets la réponse au client.

Le schéma de l'attaque apparaît clairement dans la mesure où le coût de génération d'une requête par l'attaquant est largement inférieur à celui du traitement de cette dernière.

L'efficacité de ces types d'attaque est accrue par deux facteurs. D'une part les requêtes DNS sont de petits paquets et peuvent donc être transmises en grands volumes à travers une ligne ADSL quelconque. D'autre part elles sont effectuées au-dessus d'UDP, ce qui autorise le spoofing ainsi qu'un effet de réflexion grâce aux ICMP port unreachable générés par les destinataires des réponses.

3 Effets de levier

Compte tenu des performances actuelles des serveurs et de la généralisation des techniques de répartition de charge et de haute disponibilité, il est quasiment impossible de provoquer un déni de service en « one-to-one ». Il est donc souvent nécessaire de trouver un moyen d'appliquer un effet multiplicateur à l'attaque initiale.

3.1 Zombies

L'effet de levier qui vient immédiatement à l'esprit est la technique des dénis de services distribués, mise en exergue début 2000 avec des outils tels que TFN, Trinoo ou encore Stacheldraft. L'effet est évident, un seul système contrôle ses milliers de zombies. Quelques milliers de paquets de commande vont alors provoquer un flux ininterrompu de requêtes légitimes sur un serveur Web. Les conséquences sont connues.

L'inconvénient dans ce cas est la nécessité de travailler en deux temps :

1. Mass-hacker les systèmes destinés à héberger les zombies,
2. Lancer les ordres.

A la deuxième étape le paquet de commande peut être bloqué par un outil de détection ou de filtrage. Par conséquent l'évolution consiste à automatiser le lancement des commandes dès la corruption des systèmes relais. Cette technique a été mise implémentée typiquement par CodeRed dont l'objectif était de faire connecter les serveurs corrompus au site Web de la maison blanche à une date précise...

Dans le même ordre d'idées les DDoS basés sur les canaux IRC comme canaux de communication. L'objectif est ici non plus d'établir une connexion directe entre le maître et les zombies, mais d'utiliser un serveur IRC (ou plutôt un canal) comme relais. Cette méthode, initiée en Juillet et Août 2001 par Knight et Kaiten, présente de nombreux avantages.

- Les commandes sont transmises d'une manière asynchrone via un flux « sortant », qu'il s'agisse du point de vue du maître ou de l'agent. Il est donc plus probable que le zombie puisse obtenir ses ordres ;
- avec le support SSL il est impossible de détecter les commandes passées, et par conséquent d'identifier le canal IRC servant de relais. De la même manière le maître est quasi-indétectable ;
- l'attaquant dispose d'une plate-forme de relais (le canal IRC) distribuée.

3.2 Tierce partie

L'usage des techniques de tierce partie permet de libérer les ressources du système attaquant, donnant à ce dernier un avantage supplémentaire.

L'étude de Naphta fournit un cas concret. L'objectif est ici d'établir une session TCP complète sur le serveur cible. L'établissement et le maintient de cette connexion étant consommateurs de ressources de part et d'autre (client/attaquant et serveur/cible) il est nécessaire de la « resetter » du côté du client sans que le serveur ne soit impacté. Ainsi à l'issue de cette connexion le client/attaquant émet une requête spécifique à destination d'un serveur tiers. Cette requête contient les paramètres de la connexion en cours (adresses, ports et numéros de séquence) et crée le paquet d'interruption (RST) approprié. Une fois reçu par le stack du client/attaquant la connexion est supprimée de la table de ce dernier alors qu'elle est toujours présente dans la table du serveur/cible.

Utilisée pour accroître l'effet des attaques par épuisement de ressources applicatives cette technique peut également être appliquée aux zombies. Dans ce dernier cas une attaque distribuée dont les milliers de sources sont rendues plus performantes grâce à un serveur tiers, s'avérera terriblement efficace !

3.3 Rebonds et réflexion

Une approche consiste à rebondir sur plusieurs systèmes afin d'obtenir un effet multiplicateur sur le nombre de paquets. Le cas d'école est la technique

de smurfing, consistant à envoyer sur un réseau conséquent un « ICMP echo request » en broadcast et dont la source a été spoofée pour apparaître comme celle de la victime. Ainsi pour chaque « ICMP echo request » émis depuis la machine attaquante n « ICMP echo reply » seront envoyés à la cible de l'attaque, provoquant une consommation de CPU importante, voire une saturation des liens réseau. La difficulté de mise en œuvre réside de nos jours dans le fait de trouver un réseau ouvert aux « pings » en broadcast sur Internet. Néanmoins cette technique reste efficace à partir d'un réseau local dont un individu souhaiterait détourner les ressources à son profit.

D'une manière plus subtile utiliser la réflexion avec comme effet de levier un ratio (volume réponse)/(volume requête) élevé est une solution aujourd'hui beaucoup plus simple à mettre en œuvre. Classiquement cette attaque est menée via le DNS. Ainsi une requête pour la résolution d'un MX fait 60 octets + nom du domaine (ex. google.com). La réponse quant à elle, sur un domaine ayant quelques serveurs mails fera plusieurs centaines d'octets. Par exemple :

```
[root@localhost root]# dig cisco.com mx
```

- requête : 70 octets,
- réponse : 545 octets,
- ratio : 7,78.

D'autres protocoles sont également appelés à fournir des effets de levier par réflexion considérables. Ainsi SIP, dont la « cible » des réponses est contenue dans les données d'un paquet « INVITE » de quelques centaines d'octets peut, en fonction de ses applications, être à l'origine d'un flux continu à destination de cette cible.

4 Moyens de protection

4.1 SYNCookies

L'objectif des syncookies est de protéger les systèmes contre les synfloods en les affranchissant de la nécessité de garder en mémoire l'ensemble des connexions en cours d'établissement. La mise en œuvre de cette technique consiste à affecter à chaque nouvelle demande de connexion non plus un numéro de séquence complètement aléatoire mais une valeur prenant en compte certaines caractéristiques de la connexion en cours, en intégrant toujours une notion d'aléa indispensable dans la protection contre les attaques de type spoofing.

Cette valeur, appelée le SYNCookie, est construite à partir des éléments suivants.

1. Caractéristiques IP de la connexion (adresses IP source et destination) ;
2. Caractéristiques TCP de la connexion (ports source, cible et numéro de séquence contenu dans le SYN du client) ;
3. Un secret garantissant l'impossibilité de reconstruire le syncookie ;
4. Un compteur temporel permettant d'évaluer « l'age » d'un syncookie.

L'ensemble de ces valeurs est utilisé dans la construction d'une empreinte irréversible ne permettant pas de déterminer à l'avance le numéro de séquence qui sera utilisé.

A la réception d'un paquet SYN de demande de connexion le serveur construit le syncookie et le transmet au client avec l'accusé de réception (paquet SYN+-ACK). Le serveur ne garde aucune information en mémoire. Lors de la dernière phase, le client accuse réception du syncookie en l'incrémentant de 1. Le serveur reconstruit le syncookie en prenant quelques valeurs acceptables du compteur temporel et le compare avec l'accusé de réception reçu. Si une des valeurs calculées correspond, la session est établie et le TCB correspondant est construit. Le cas échéant la session n'est pas prise en compte.

4.2 Anti-Spoofing par TTL

Comme nous avons pu le voir de nombreuses attaques sont possibles en spoofant l'adresse IP source de l'attaquant. Dans certains cas cette technique permet également d'accroître l'efficacité de l'attaque. Dans la mesure où il ne s'agit pas ici de tentative d'usurper une adresse du réseau interne, la connaissance des réseaux connectés aux différentes interfaces de l'équipement de filtrage n'est pas pertinente.

En revanche il est envisageable de s'appuyer sur une technique de calcul du TTL pour les paquets susceptibles d'être spoofées. Le principe, relativement simple, est le suivant. Un paquet prétend venir d'une adresse A avec un TTL de 54. Un simple calcul permet de déduire qu'il a effectué 10 sauts avant d'arriver à destination. Si un ping de cette adresse IP nous revient avec un TTL caractéristique de 3 ou 18 sauts cela signifie clairement que le paquet initial a été spoofé. Bien entendu la génération d'un ping pour chaque paquet reçu est exclue et il est nécessaire de construire une cartographie des réseaux et de leur « distance ».

S'il est indispensable de prendre une marge dans la mesure où les routes peuvent varier, il est toutefois envisageable de s'appuyer sur cette technique pour réduire de manière significative l'effet d'attaques de type SYNFloods ou épuisement par anomalies.

En outre la généralisation de ce type de technique sur les réseaux permet d'éviter que les serveurs ne soient utilisés pour une attaque par réflexion.

4.3 Echantillonnage

Les attaques par dénis de service se basent principalement sur des séquences caractéristiques et des volumes importants. Leur détection peut par conséquent s'effectuer à deux niveaux.

Au premier niveau il est possible de définir des combinaisons de critères caractéristiques d'anomalies. De manière triviale un paquet contenant un flag SYN ne peut contenir un flag RST. Un peu plus loin dans la réflexion, à l'issue de l'établissement d'une session TCP sur le port 80, le paquet suivant doit être

une commande GET. Plus compliqué encore la présence récurrente de paquets ACK+PUSH est synonyme de tentative de déni de service.

La création de telles « signatures » n'est pas particulièrement en complexe. En revanche leur application sur des volumes de plusieurs centaines de Mbps pose un réel problème de performance.

Par conséquent, et compte tenu du fait que ces attaques n'ont d'effet que sur de gros volumes, il est possible de mettre en place un mécanisme d'échantillonnage, basé sur l'analyse d'un paquet sur « n ». A l'issue de la détection de « m » occurrences de l'événement dans un temps imparti, la signature en question est activée et appliquée à l'ensemble des paquets.

Cette technique présente l'avantage de réduire considérablement l'utilisation de CPU tout en présentant un taux de détection particulièrement fiable comme le prouve le calcul suivant.

Soit :

- n, le taux d'échantillonnage,
- N, le nombre total de paquets par seconde,
- s, le délai au cours duquel m occurrences doivent être détectées,
- E, le nombre d'échantillons relevés au cours de s.

$$E = N \times n \times s$$

- A, le pourcentage de trafic d'attaque par rapport au trafic légitime,
- p, la probabilité de ne pas détecter une attaque au cours de chaque échantillonnage,

$$p = 1 - A,$$

- m, le nombre d'occurrences pour identifier une attaque au cours de s avec $m < E$.
- P, la probabilité qu'une attaque ne soit pas détectée à l'issue de s,

$$P = p^(E - m)$$

Avec des données réelles nous obtenons les résultats suivants :

- $n = \frac{1}{1000}$,
- $N = 150.000$ pps,
- $s = 0,1$ seconde,
- $A = 80\%$,
- $m = 10$,

$$P(s) = 3,2E - 4.$$

Ainsi après l'analyse de $E = 15$ paquets il y a 0,0032% de chances que l'attaque ne soit pas détectée, ce avec une probabilité de faux-positifs particulièrement faible dans la mesure où il est peu probable que dans le cas du dysfonctionnement d'une application ou d'un équipement réseau 10 paquets soient transmis en 1/10è de seconde.

4.4 Décodage des protocoles

La compréhension des mécanismes de communication entre les systèmes est un élément particulièrement crucial pour la détection des dénis de service.

Au niveau des couches 3/4 le mécanisme est connu et, normalement, implémenté de manière appropriée par les équipements de filtrage. Il s'agit du stateful inspection. La mise en œuvre de ce type de technologie permet en particulier de se protéger des attaques par réflexion. Ainsi un SYN-ACK ne correspondant à aucune session de la table des connexions sera systématiquement rejeté.

Il n'en reste pas moins que certains cas un peu plus complexes peuvent ne pas être traités par des moteurs stateful peu matures. Les protocoles de contrôle en particulier, qui utilisent un autre canal sont particulièrement problématiques. Le plus trivial d'entre eux, ICMP peut s'avérer être un véritable cauchemar, car si la gestion des ICMP echo reply en réponse aux ICMP echo request est une opération banale, il n'en n'est pas de même pour le reste des messages ICMP.

Prenons le cas des ICMP destination unreachable. Ces messages peuvent être émis dans le cas d'hôtes et de réseaux inaccessibles, de ports fermés, de protocoles non supportés, de fragmentation impossible quand le flag DF (Don't Fragment) est positionné et quand le source routing n'est pas autorisé. Si les 3 premiers cas sont simples à traiter les autres impliquent que la quasi-totalité des champs d'en-tête IP soient gardés dans les tables d'états, ce afin de vérifier qu'effectivement tel flag était positionné ou que le source routing était précisé dans les options IP. Il en est de même pour les messages de type Time Exceeded, Source Quench et Parameter Problems.

Au niveau applicatif le suivi est également indispensable, ce pour des raisons similaires. Encore une fois certains modèles sont faciles à traiter, en particulier dans le cas de protocoles symétriques tels que HTTP ou DNS, où requêtes et réponses utilisent le même canal et sont « relativement » simple à décoder.

Deux autres cas de figure sont beaucoup plus complexes à gérer. Le premier concerne le « décodage ». En effet certains protocoles (RPC par exemple) ne sont que des cadres qui permettent la transmission d'information. Charge à l'application supportée par ce protocole de les comprendre et de les interpréter. Le suivi d'une communication nécessite par conséquent de connaître parfaitement le fonctionnement de l'application, ce qui n'est pas toujours évident compte tenu de leur nombre et de leurs spécificités parfois liés au système d'exploitation. Dans le même ordre d'idée les exceptions et interprétations non-conformes aux RFCs peuvent ajouter une complexité considérable aux opérations d'analyse et de détection.

Prenons le cas d'école d'une requête HTTP. Le format normal est le suivant :

```
<COMMANDE> <PATH> HTTP/<VERSION>
```

Soit par exemple :

```
GET /index.asp HTTP/1.1
```

Il est intéressant de remarquer que les serveurs web comprennent une requête ne présentant pas le dernier argument (protocole/version), comme le prouve l'exemple suivant.

```
C:\tools>nc 192.168.202.106 80
GET /index.asp
<html>
 <head><title>Hello</title></head>
 <body><h1>hello</h1></body>
</html>
```

Le second cas de figure concerne les applications utilisant des canaux différents pour le contrôle et le transfert de données. Ainsi une application se basant sur RTP (Real Time Transfer Protocol) pour le transfert de flux en streaming pourra s'appuyer sur SIP (Session Initiation Protocol) pour l'établissement de la connexion. La légitimité du trafic en streaming ne pourra être validée qu'après analyse d'une éventuelle requête SIP INVITE. A défaut d'une telle analyse soit le flux légitime sera coupé soit une attaque basée sur le spoofing dans l'en-tête SIP passera comme une fleur...

4.5 Limitation des sessions

A partir du moment où le protocole applicatif est décodé, il devient possible non seulement de suivre les sessions mais également d'établir un certain nombre de critères tels que le nombre de sessions établies simultanément par source, le volume et la typologie des flux à l'intérieur de la session... Cette analyse est d'autant plus fine que l'application en question est décodée en détail. Le suivi d'une session HTTP basé sur un cookie, voir le positionnement à posteriori d'un cookie par le système de protection sont des opérations particulièrement intéressantes pour l'identification de trafics.

A partir de là, il est possible de limiter le trafic selon différents critères, le plus utilisé étant généralement le nombre de sessions simultanées par source. En effet, dans le cas de zombies ou flashmobs, les « agents » établissent plusieurs centaines, voir milliers, de connexions simultanées. Ces connexions étant parfaitement légitimes il est quasiment impossible de les bloquer sans bloquer des flux inoffensifs. En limitant le nombre de sessions simultanées l'impact de l'attaque est considérablement réduit et les flux légitimes ne sont pas affectés.

Un autre avantage de cette technique est sa capacité à protéger des DDoS au niveau du réseau final, et non plus de l'opérateur. En effet, un des principaux effets des DDoS est la saturation des liens vers l'opérateur du au volume important de données transférées en réponse aux requêtes (HTTP en général). La limitation du nombre de sessions simultanées réduit de facto ce volume et par conséquent l'impact sur le « tuyau ».

4.6 Mise en œuvre et mode opératoire

Si les SYNCookies fournissent une réaction native consistant à ne pas établir de session TCP, les mécanismes d'anti-spoofing et d'échantillonnage ne sont que des techniques de détection. En outre les aspects performance restent cruciaux compte tenu des débits importants devant être traités.

Une mise en œuvre permettant de réduire au maximum l'utilisation de CPU suivrait par conséquent le schéma suivant :

- Anti-Spoofing par TTL. A l'issue de la vérification trois catégories de trafic sont définies : sûr, suspect et malicieux. Le trafic sûr, associé à des réseaux maîtrisés n'est plus analysé. Un trafic est qualifié de suspect dans la mesure où les paquets ne proviennent pas d'un réseau sûr et/ou que le TTL n'est pas exactement celui attendu. Ce trafic est passé à l'étape suivante. Le trafic malicieux, correspondant au reste des flux est systématiquement rejeté.
- SYNCookies. Les SYNcookies sont mis en place. Les attaques de type SYNFlood et les tentatives de spoofing restant sont « nettoyées ».
- Echantillonnage. L'échantillonnage et l'analyse des flux restant permet d'éliminer le reste des attaques, à savoir les attaques par réflexion sur UDP et les attaques par épuisement de ressources depuis des zombies ou supportées par des serveurs tiers. La réaction peut être bien entendu le blocage des flux, mais, afin de réduire encore une fois les possibilités de faux-positifs, il peut également être pertinent de se contenter de réduire la bande passante allouée à ce trafic. En cas d'attaque celle-ci ne disposera plus d'assez de bande passante pour avoir un effet notable et en cas de trafic légitime, ce dernier ne sera pas coupé. Dans le même ordre d'idées les zombies et flashmobs pourront voir leur efficacité considérablement réduite en limitant le nombre de sessions simultanées à destination de certaines ressources (service web, mail...).

En termes de modes opératoires, deux schémas types sont généralement proposés.

Le premier, et le plus simple, consiste à mettre les équipements de protection en coupure sur les liens à protéger. Cette approche garantit en général une mise en place relativement simple et n'implique aucun impact sur l'infrastructure logique du réseau. En revanche les problématiques de performances sont vitales et la sensibilité aux faux positifs extrêmement élevée.

Le second schéma défini une séparation entre les opérations de détection et de « nettoyage ». Une sonde détecte un déni de service et effectue un reroutage BGP ou un MPLS shunt à destination d'un système en charge de la protection. Dans certains cas (blackhauling) le système de nettoyage est `/dev/null`. De nos jours, (...) sont mis en place des « machines à laver » dont la fonction est l'élimination des trafics malicieux. Si ce mode d'implémentation permet de s'affranchir (dans une certaine mesure) des problématiques de performances et réduit l'impact des faux positifs (uniquement sur une partie du trafic déjà considéré comme suspect), elle présente l'inconvénient de n'être sérieusement envisageable que sur des réseaux d'opérateurs et d'impacter les opérations de routage. Des tests intensifs doivent par conséquent être effectués avant le déploiement.

Le mode de détection et les critères de routage (dans le second schéma) sont particulièrement importants, en particulier lorsque la prévention est effectuée par blackhauling. En effet, une décision de routage basée sur la source ne pourra pas être efficace en cas d'attaque spoofée. Qui plus est elle fournira (blackhauling)

un déni de service intégré, des pans entiers de réseaux pouvant être expédiés vers nulle part car identifiés comme sources d'attaques. Basée sur la destination, la décision de routage devient plus pertinente à deux titres. D'une part, que les attaques soient spoofées ou distribuées, c'est au niveau de la cible que les flux vont converger. Ainsi les critères de détection (volume, nombre de paquets...) seront plus rapidement remplis. D'autre part, si le blackhauling génère de facto un dénis de service sur la cible (tous les trafics à destination de cette dernière sont systématiquement rejetés), le reste de l'infrastructure reste opérationnel et ne subit pas d'effets de bords.

5 Conclusion

Les dénis de service existent toujours. Leur efficacité a été accrue par l'augmentation de la puissance des ordinateurs personnels et de la capacité des moyens d'accès à Internet. En parallèle l'impact de telles attaques a évolué au point de mettre en péril l'activité des entreprises.

Les solutions quant à elles existent, mais sont déployées de manière disparate sur différents points du réseau et sans coordination entre les composants. Qui plus est, ces solutions sont rarement adaptées, en termes de performance, aux attaques de grande ampleur et dont les trafics sont atypiques.