Séminaire de détection des intrusions
Le déni de service (DoS)

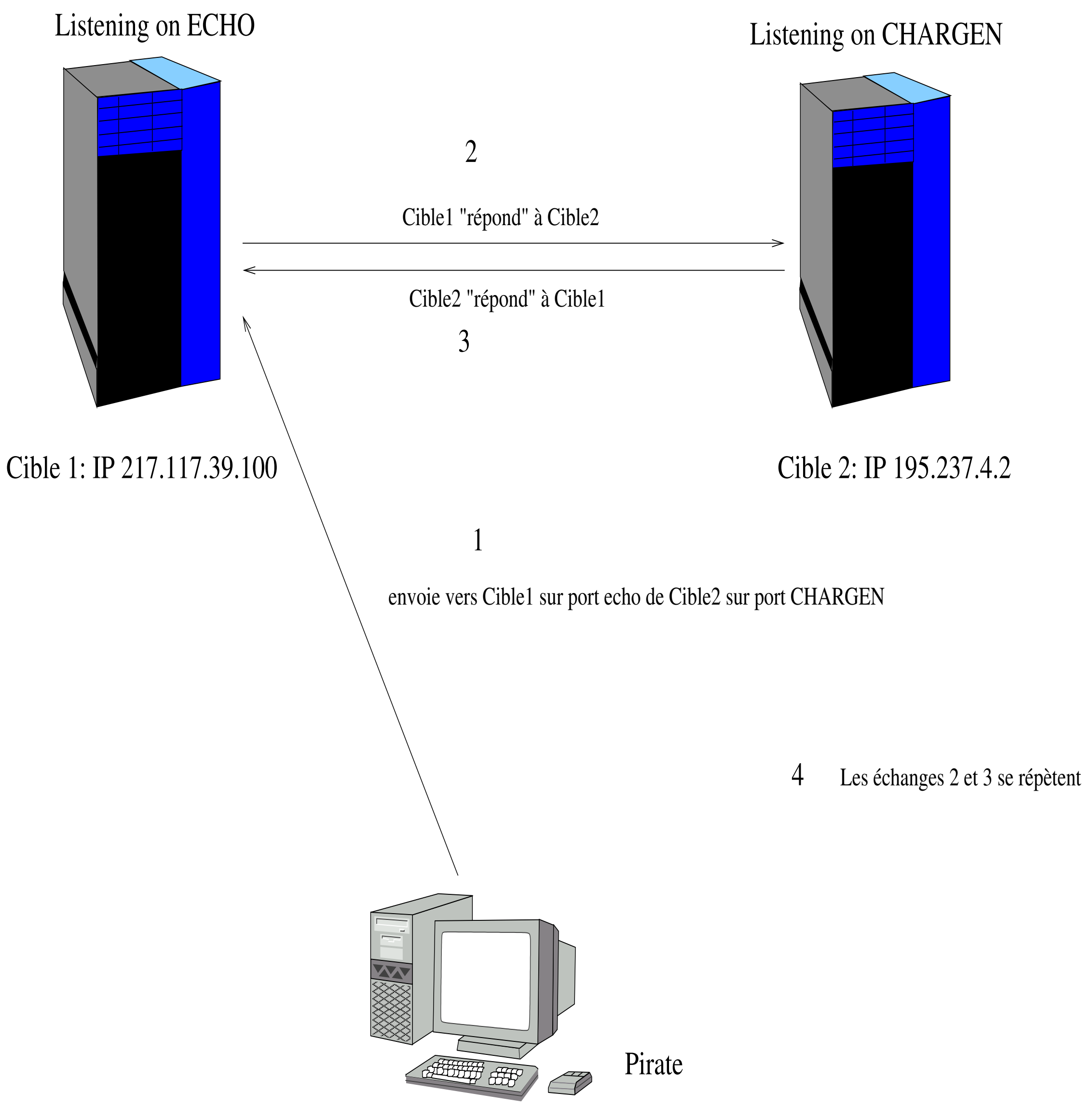
Boniver Christophe
Herbiet Laurence
Joseph Benoit
Seronveaux Xavier

Les attaques DoS

Elles consistent à paralyser temporairement (rendre inactif pendant un temps donné) des serveurs afin qu'ils ne puissent être utilisés et consultés.

Le but d'une telle attaque n'est pas de récupérer ou d'altérer des données, mais de nuire à des sociétés dont l'activité repose sur un système d'information en l'empêchant de fonctionner.

Exemple de DoS



Première attaque : Attaque sur Snort

- Création d'un déni de service sur l'outil Snort (v 1.8.3) à l'aide d'un paquet ICMP envoyé par *ping*
- La commande utilisée est :

  ping -c 1 -s 1 [host]

  – L'option -c 1 indique qu'il faut s'arrêter après avoir envoyé (et éventuellement reçu) 1 paquet ECHO_RESPONSE.
  – L'option -s 1 spécifie la taille (byte) des données à être envoyées. Ici, la taille du paquet envoyé est de 1 byte.

Rappel :

- ICMP (Internet Control Message Protocol) : protocole de contrôle et de gestion dans la couche réseau.
- Chaque type de messages ICMP sont encapsulés dans un datagramme IP

Format de l'en-tête du datagramme IP

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|HeadLen|     T.O.S     |     Longueur Totale
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        | Flags |     Dép_fragment
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Durée de vie  |   Protocole   |   Checksum sur le Header
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Adresse Source
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Adresse de Destination
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

L'en-tête IP contient donc 20 octets

Format de l'en-tête ICMP

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Identifier          |        Sequence Number
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

L'en-tête ICMP contient donc 8 octets

Données :

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Data      |
+-+-+-+-+-+-+-+-+
```

La taille des données est de 1 octet.

En prenant la taille de l'en-tête du datagramme IP et ICMP plus 1 octet de data, nous avons un paquet de taille égale à 29 bytes.

Exemples avec "tcpdump" :

On effectue : *ping -c 1 -s 1 localhost*
Résultat de TCPDump.

```
23:22:16.069711 localhost > localhost: icmp: echo request
4500 001d 01e2 0000 4001 7afc 7f00
0001 7f00 0001 0800 76fe 8101 0000 00

23:22:17.067666 localhost > localhost: icmp: echo request
4500 001d 01e4 0000 4001 7afa 7f00
0001 7f00 0001 0800 75fe 8101 0100 00
```

Analyse des paquets :
Les données correspondent bien aux champs décrits ci-dessus :

```
Offset:
             0    2    4    6    8    A    C    E

        0    4500 001d 01e2 0000 4001 7afc 7f00 0001

        1    7f00 0001 0800 76fe 8101 0000 00
```

Le pattern "7f00 0001" (c'est à dire "127.0.0.1") à l'offset 0Ch et 10h, correspond bien aux champs d'adresse source et destination.

```
Offset:
            0    2    4    6    8    A    C    E

            0    4500 001d 01e2 0000 4001 7afc 7f00 0001
            1    7f00 0001 0800 76fe 8101 0000 00
```

La valeur 08h à l'offset 14h, correspond au type de message, c'est à dire : Echo Request (voir decode.h).

On remarque à l'offset 1Ah que dans le premier paquet, la valeur est 00h et dans le second, la valeur est 01h, et on peut donc supposer qu'il s'agit du numéro de séquence.

Et le bug de Snort dans tout ça ?

Analysons le code de snort ("decode.h" et "decode.c") :

- Lorsque Snort reçoit le paquet ICMP, il appelle d'abord la fonction "DecodeIP" en lui passant trois arguments :
  - pkt : pointeur vers le paquet reçu
  - len : longueur du paquet
  - p : pointeur vers une structure de décodage

- Après plein de tests sur les checksum,fragments, ... il appelle la fonction "DecodeICMP" en lui passant également trois arguments :
  - pkt + hlen : pointeur vers le paquet ICMP
  - ip_len : longueur du contenu du paquet IP, c'est à dire la longueur du paquet ICMP
  - p : pointeur sur la structure de décodage

- DecodeICMP refait des tests sur la longueur du paquet ICMP, checksum, ... Ensuite, il définit la longueur des données comme la taille du paquet ICMP moins la taille du Header, et définit un pointeur vers les données, c'est à dire, le pointeur du paquet ICMP plus la taille du Header.

```
p->dsize = (u_short) (len - ICMP_HEADER_LEN);
p->data  = pkt + ICMP_HEADER_LEN;
```

- Ensuite survient le bug, ...
  Il teste le type de message, et dans le cas d'un "echo", ou "echo reply", il décale de nouveau le pointeur vers les données 4 bytes plus loin et réduit la taille des données de 4 bytes.

```
p->dsize -= sizeof(struct idseq);
p->data  += sizeof(struct idseq);
```

Or dans l'exemple précédent, la taille du paquet ICMP était de 9 bytes...

$\Rightarrow$ si on saute le Header, il reste 1 byte, et si on saute encore 4 bytes, on est hors du buffer
$\Rightarrow$ Segmentation fault, core dumped, ...

Pourquoi Snort n'accepte pas des données de moins de 4 bytes ?

Cette vulnérabilité provient du fait que Snort considère que dans certains cas,notamment celui du message "REDIRECT", il est intéressant d'avoir les 4 derniers bytes du header ICMP dans les données.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Gateway Internet Address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Internet Header + 64 bits of Original Data Datagram      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Si une attaque était réalisée avec des messages de ce type, il peut effectivement être intéressant d'avoir dans les données le champ "Gateway Internet Address"
C'est pour cela qu'il faut considérer que l'en-tête ICMP fait 4 bytes, et que dans le cas d'un "ECHO", il saute les 4 bytes suivants.

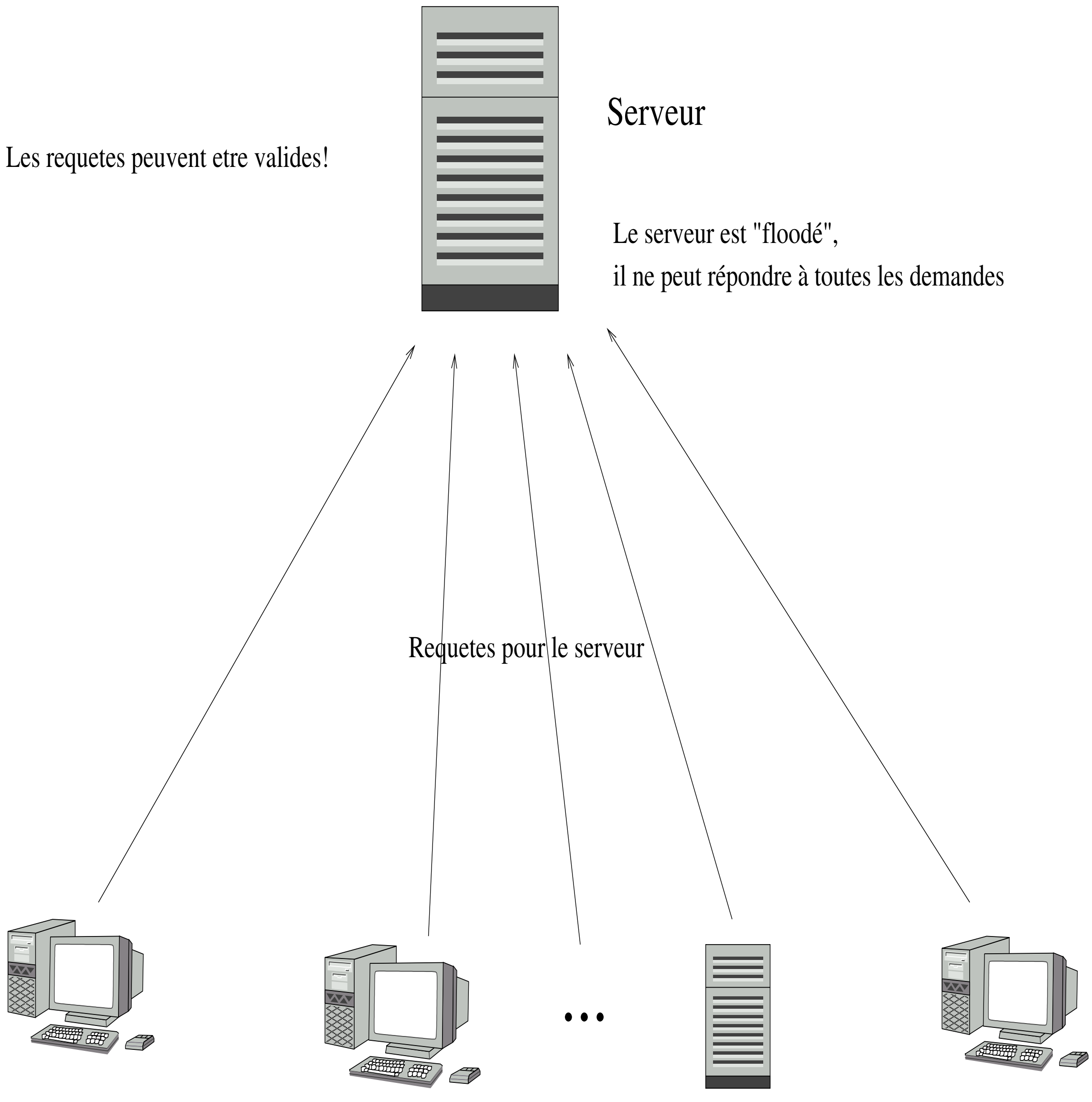
Détection de ce type d'attaque

Pour détecter ce type d'exploit, il faut regarder si on a bien un paquet ICMP de type "ECHO REQUEST" ou "ECHO" dont la taille des données est inférieure ou égale à 4 bytes.

Les attaques DDoS

Tout comme les attaques DoS, les DDoS ont pour but de mettre à genou une cible et non d'obtenir des informations contenues sur celle-ci.
La technique des DDoS à la particularité d'être distribuée càd que l'attaque est réalisée non plus par une machine mais par une multitude.
La première opération d'une attaque par refus de service distribué consiste à obtenir un accès administrateur sur le plus grand nombre de systèmes possible. Ensuite, il reste à l'assaillant à charger ses logiciels de refus de service distribué et de les exécuter.

Schéma d'un DDOS



Deuxième attaque

Cette attaque est un DDOS contre un serveur WEB.
Elle (ou une attaque très similaire) a été portée contre le site du World Economic Forum. Cette attaque a été orchestrée par un mouvement zapatiste. Les liens vers les sources et pages de l'attaque étaient disponible seulement pendant l'attaque. D'après leurs dires, plus de 100.000 personnes se seraient raliés à eux pour porter le coup fatal...

Description de l'attaque

L'attaque que nous avons pu récupérer permet d'attaquer sur deux points :

- flood : L'applet recharge une même page un grand nombre de fois par minute. (point principal)
- spam : l'applet permet également de laisser un "message" sur le serveur cible. Ceci est possible car les serveurs journalisent les requêtes et plus particulièrement celles retournant un code d'erreur 404 ("page not found") (point secondaire)

Mise en oeuvre de l'attaque

Ici, les initiateurs de l'attaque n'ont même pas eu à se donner le mal de prendre le contrôle d'un parc de machines. L'attaque est lancée par une applet que l'on active en chargeant une certaine page avec n'importe quel navigateur supportant le Java. L'utilisateur arrive sur une page html où on lui indique la marche à suivre. Il faut désactiver quelques options du navigateur pour que l'attaque soit optimale. Il peut rentrer un message personnel pour la partie spam de l'attaque... C'est tout !

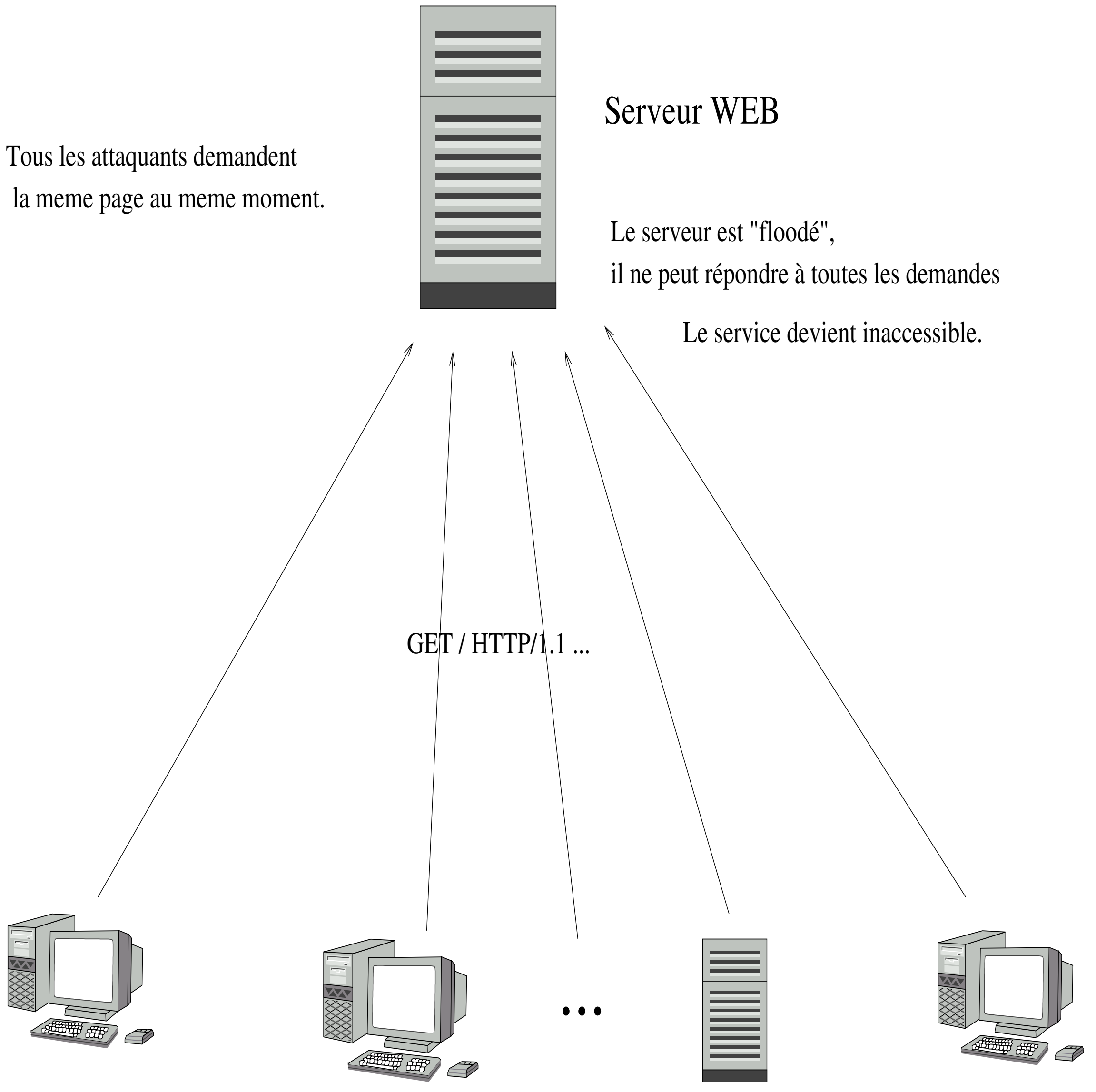
Pour augmenter l'ampleur de l'attaque, le code source de l'applet a été rendu plublic durant la période de l'attaque.

Il faut bien être conscient que ce type d'attaque demande un nombre conséquent de participants pour avoir de l'effet. Plus l'organisme ciblé possède de ressources, plus ce nombre devra être grand.

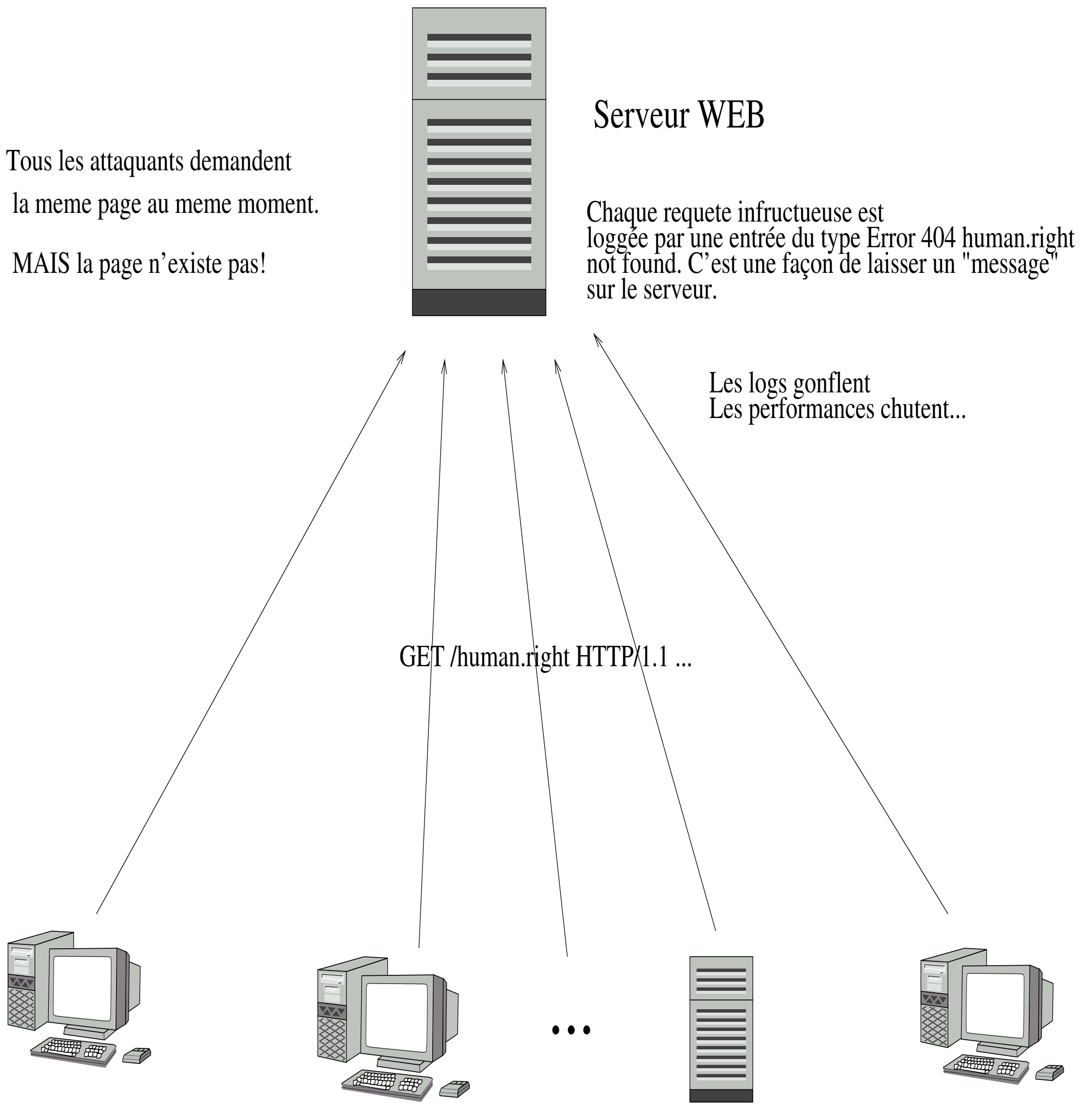
Synchronisation des attaquants

Ici aussi la méthode employée est d'une simplicité enfantine. La date de l'attaque est indiquée sur le site. Libre à qui veut de s'y joindre...

Attaque par flooding



Effet secondaire/de bord : Attaque SPAM



Causes du déni de service

Elles sont relativement évidentes :

- La première raison est que la machine est simplement incapable de répondre à l'énorme quantité de requêtes, valides ou non, qu'elle reçoit.
- La seconde est un peu moins immédiate. Elle est causée en grande partie par la partie spam de l'attaque. Le serveur journalise un grand nombre d'informations sur les requêtes, surtout si la dite requête échoue pour une raison ou une autre. L'attaque par spam vise à laisser un message dans les logs par l'envoie d'une requête non valide contenant le message. La gestion de ces logs par le serveur devient de plus en plus fastidieuse au fur et à mesure que les logs gonflent.

Détection de l'attaque

Tout le problème est de détecter ce type d'attaque. Le but du serveur est de gérer des requêtes qu'elles soient valides ou non.

La seule solution qui nous vient à l'esprit est de prendre un compteur sur les urls demandées et un autre sur le nombre d'erreurs 404 retournées. En effet, si un nombre "raisonnable" de requêtes sur une même page dans un laps de temps "raisonnable", on peut supposer qu'une attaque par DDOS est en cours.

Il est bien évident que cette solution risque de très vite devenir coûteuse.

Un moyen simple de protéger le serveur est de limiter le nombre de requêtes ou de connexions arrivant au serveur.