Denial of Service and Distributed Denial of Service

Laurence Herbiet
Christophe Boniver
Benoit Joseph
Xavier Seronveaux

DoS

- Rappels
- Signature de l'attaque

DDoS

- Rappels
- Signature de l'attaque

Denial of Service

- Rappels
  - Qu'est ce qu'un DoS ?
  - Descritpion de l'attaque
  - Conséquence de l'attaque
  - Cause de l'attaque
- Signature de l'attaque
  - Création d'une règle sous snort

Qu'est-ce qu'un 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'un telle attaque n'est pas d'obtenir 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.

Descritpion de l'attaque

Attaque

Création d'un déni de service sur l'outil Snort (version 1.8.3) à l'aide d'envoi de trame ICMP (ECHO REQUEST)

Pour réaliser l'attaque, il suffit d'envoyer une trame ICMP de type ECHO REQUEST dont la taille des données est inférieure à 4 bytes.

Comment ?

En utilisant la commande ***ping*** :

ping –s 1 stallman

L'option –s 1 spécifie que l'on ne transmet que 1 byte de donnée.

Conséquence de l'attaque

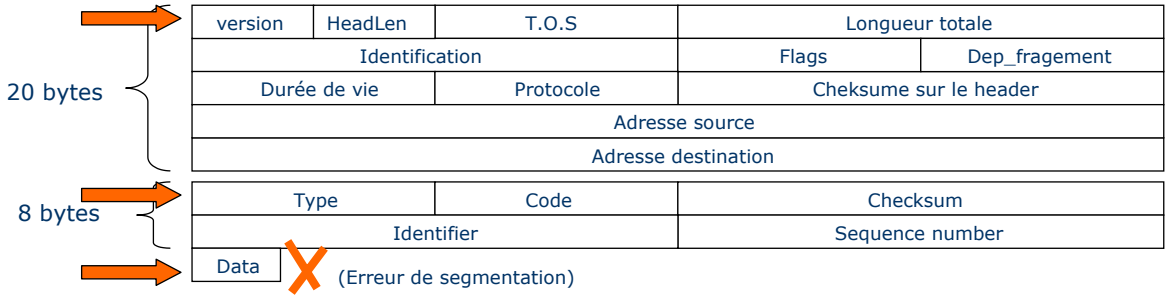
Suivant les règles utilisées dans la configuration de Snort, le résultat de l'attaque diffère :

- Soit Snort se crash (Erreur de segmentation)
- Soit Snort affiche à l'écran une trame dont le contenu est la mémoire de la machine attaquée.

⇒ Cela est dû à l'interraction des règles utilisées dans Snort.

Cause de l'attaque

Pacquet ICMP (encapsulé dans un datagramme IP)



1. Snort initialise un pointeur sur le paquet reçu ainsi qu'une variable contenant la longueur du paquet.

2. Snort déplace le pointeur vers l'en-tête ICMP tout en modifiant la longueur du paquet qui devient la taille du paquet ICMP

3. Pour finir, Snort déplace le pointeur vers les données et dans le cas d'un ECHO, ou ECHO REPLY, il décale de nouveau le pointeur vers les données 4 bytes plus loin.

Signature de l'attaque

- Détecter que l'on a une trame ICMP de type ECHO REQUEST
- Vérifier que les données ont une taille inférieure à 4 bytes

Les paquets étant décodés par Snort, les valeurs renvoyées par celui-ci sont erronées.
Nous utiliserons cette particularité lors de la création de la règle pour détecter l'attaque.

Création d'une règle sous Snort

- Lorsque Snort décode une trame ICMP de *n* (<=4) bytes, il indique une *payload* supérieur à 65535-*n* bytes. Cette valeur peut être obtenue dans les règles à l'aide de *dsize*.
- La règle verifira si on a bien une trame ICMP de type ECHO REQUEST(type = 8) et que la payload est supérieure à 65531 bytes.

```
alert icmp $EXT_NET any -> $HOME_NET any
(msg:''ICMP less four bytes''; itype: 8; dsize: > 65531; )
```

- Pour détecter la règle dans le cas d'une version patchée, il suffit d'appliquer la règle suivante :

```
alert icmp $EXT_NET any -> $HOME_NET any
(msg:''ICMP less four bytes''; itype: 8; dsize: <4; )
```

DoS

- Rappels
- Signature de l'attaque

DDoS

- Rappels
- Signature de l'attaque

Distributed Denial of Service

- Rappels
  - Qu'est ce qu'un DDoS ?
  - Description de l'attaque
  - Conséquence de l'attaque
- Signature de l'attaque
  - Détection des erreurs 404
  - Création d'un compteur sous Snort
  - Création d'une règle sous snort

Qu'est ce qu'un DDoS ?

- Les DDoS ont le même but que les DoS : mettre à genou une cible.
- La principale différence est que l'attaque est distribuée sur une multitude de machine

Première étape : se rendre maître de plusieurs machines

Deuxième étape : lancer l'attaque à partir de ses machines

Description de l'attaque

- Attaque DDoS contre le serveur web du World Economic Forum.
- Propagée par un mouvement Hacktivist avec une date de synchronisation.
- Télechargement volontaire de l'applet utilisée pour cet exploit.

➡ chargé par environ 100.000 personnes.

➡ pas de prise de contrôle des machines de la part des Hacktivists.

Effet de l'attaque

L'attaque permet de nuire au serveur de deux manières différentes :

- Flood : recharger une page invalide sur le serveur une grand nombre de fois par minute
- Spam : laisser un message d'erreur sur le serveur. Possible car journalisation des requêtes par les serveurs (particulièrement les erreurs 404)

Conséquence de l'attaque

➡ La machine est incapable de répondre à l'énorme quantité de requêtes.

➡ Vu la journalisation des erreurs 404 sur le serveur, les fichiers de logs gonflent jusqu'à saturer la mémoire du serveur.

Signature de l'attaque – Essais/erreurs

Solutions :

- Piste des compteurs
  - Essais - Erreurs
  - Solutions retenues
- Piste des applets
  - Essais - Erreurs
  - Solutions retenues

Signature de l'attaque – Piste des compteurs

- Compter le nombre d 'erreurs « 404 »
  - Serveur : nombre d 'erreurs 404 retournées par client pendant un laps de temps
  - Client : nombre d 'erreurs 404 reçues par utilisateur
- Compter le nombre de requêtes par URL
- Compter le nombre de requêtes (ciblées) par un utilisateur

Réalisation de la solution

- Réalisation d'un préprocesseur qui compte le nombre d'erreurs 404 par utilisateur.
- S'adapte aussi bien à un serveur qu'à des clients

preprocessor http_404: -src -to:60 -nb:10 -ti:60

- Gestion par liste circulaire doublement liée
- Vérification sur base d'un seuil et d 'un intervalle de temps selon la formule :

*Count=Count*exp(K1*(Now-Time))+K2;*

Amélioration du préprocesseur

Le préprocesseur pourrait s'adapter au cas des requêtes valides

-> compter le nombre de requêtes effectuées par client vers certaines pages particulières (.html, .php, .jsp, .jpg, …)

Signature de l'attaque – Piste des applets

- Insertion de valeurs particulières dans l 'en-tête HTTP ?
- Détection de l'applet sur une page html : détection de la balise <applet> et de son nom
- Détection de mots particuliers dans le fichier « .class »
- Détection du chargement d 'applet à partir de sites suspects

Réalisation de la solution

- Détection de la présence de l 'applet

-> le nom de l'applet :

```
alert tcp any 80 -> $HOME_NET any
(msg:"Flooding Applet";
      content:"<applet"; nocase;
      content:"zapsfloodnetpublic1.class"; nocase;)
```

-> les paramètres de l'applet :

```
alert tcp any 80 -> $HOME_NET any
(msg:"Flooding param";
      content: " targetUrl "; nocase;
      content:"evade";  nocase;)
```

Réalisation de la solution (2)

- Détection du chargement d'applet à partir de sites suspects

```
alert tcp $HOME_NET any -> 192.168.1.44/32 80
(msg:"flooding server"; uricontent:".class";nocase;)
```

QUESTIONS ?