LinuxFocus article number 292
http://linuxfocus.org

IDS - Systèmes de Détection d'Intrusion, Partie I

par Klaus Müller
<Socma(at)gmx.net>

L´auteur:
Klaus Müller alias 'Socma' est encore étudiant et s'investit dans la programmation et la sécurité de Linux.

Traduit en Français par:
Georges Tarbouriech
<gt(at)linuxfocus.org>

Résumé:

Voici le premier article d'une série sur les Systèmes de Détection d'Intrusion. La première partie présente non seulement les architectures IDS mais les attaques classiques contre ces systèmes sont également discutées et analysées.

Introduction

Avant de commencer, voici quelques explications afin d'éviter les malentendus : IDS signifie Système de Détection d'Intrusion (Intrusion Detection System) et au cours de l'article je me réfère aussi aux IDS en tant que systèmes capables de détecter des intrusions et de prendre des mesures de protection (même si ces mesures sont optionnelles). Je ferai d'abord un tour d'horizon des différents types d'IDS afin de définir plusieurs tactiques. Dans la dernière partie j'aborderai des programmes comme Snort, bofh... et mon projet COLOID.
Au préalable, je propose quelques définitions de termes qui ne seront pas directement expliqués dans le texte :

- faux positif = une alerte pour une attaque qui n'a pas eu lieu
- faux négatif = l'IDS ne détecte pas et ne filtre pas l'attaque

Vue d'ensemble d'un IDS

Dans ce chapitre, l'idée consiste à présenter les différents types de Systèmes de Détection d'Intrusion ainsi que leurs avantages et inconvénients. Mais d'abord, voici quelques mots de présentation sur le rôle et les intentions de ces systèmes. Un IDS est censé observer l'activité sur un hôte donné ou sur un réseau donné. Selon plusieurs méthodes que je présenterai, il essaie de découvrir si la sécurité d'un hôte est menacée (s'il est "attaqué") afin de prendre les mesures de protection qui s'imposent. Puisque dans la plupart des cas, des fichiers de log sont créés, l'essentiel consiste à les analyser et à réagir à une intrusion ou à une tentative illégale d'un utilisateur d'augmenter ses droits.
Il existe trois types de champs d'activité :

- Sources d'Information
- Réponse
- Analyse

Réponse et Analyse seront traitées plus tard (respectivement dans les chapitres 6 et 4). Voyons tout de suite les différentes sortes de "Sources d'Information". En raison des multiples possibilités d'attaques d'un PC ou d'un réseau, il existe différents types d'IDS (pouvant d'ailleurs être combinés) qui varient selon l'endroit qu'ils surveillent et ce qu'ils contrôlent (les Sources d'Information).

La détection d'Intrusion basée sur l'hôte

Les systèmes de détection d'intrusion basés sur l'hôte analysent exclusivement l'information concernant cet hôte. Comme ils n'ont pas à contrôler le trafic du réseau mais "seulement" les activités d'un hôte ils se montrent habituellement plus précis sur les types d'attaques. De plus, vous remarquez immédiatement l'impact sur la machine concernée comme par exemple si un utilisateur l'attaquait avec succès. Ces IDS utilisent deux types de sources pour fournir une information sur l'activité : les logs et les traces d'audit du système d'exploitation. Chacun a ses avantages : les traces d'audit sont plus précises et détaillées et founissent une meilleure information alors que les logs qui ne fournissent que l'information essentielle sont plus petits. Ces derniers peuvent être mieux contrôlés et analysés en raison de leur taille, mais encore une fois, les deux méthodes ont des avantages et des inconvénients.

Avantages :

- vous constatez l'impact d'une attaque et pouvez donc mieux réagir
- vous découvrez plus facilement un Cheval de Troie puisque les informations et les possibilités sont très étendues.
- vous pouvez détecter des attaques impossibles à détecter avec des IDS réseau puisque le trafic y est souvent crypté
- vous pouvez observer les activités sur l'hôte avec précision

Inconvénients :

- ils ont moins de facilité à détecter les scans
- ils sont plus vulnérables aux attaques de type DoS
- l'analyse des traces d'audit du système est très contraignante en raison de la taille de ces dernières
- ils consomment beaucoup de ressources CPU

Exemples:

- Tripwire [http://www.tripwire.com/products/index.cfml]
- SWATCH [http://freshmeat.net/redir/swatch/10125/url_homepage/swatch]
- DragonSquire [http://www.enterasys.com/ids/squire/]
- Tiger [http://freshmeat.net/redir/tiger-audit/30581/url_homepage/tiger]
- Security Manager [http://www.netiq.com/products/sm/default.asp]

La Détection d’Intrusion Réseau (NIDS)

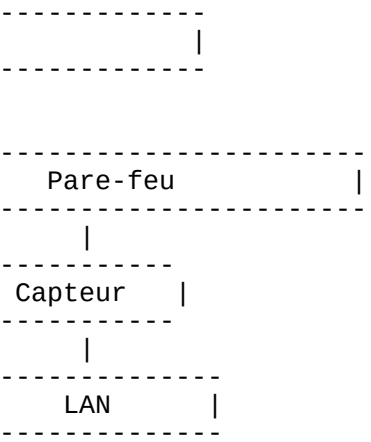
Le rôle essentiel d’un IDS réseau est l’analyse et l’interprétation des paquets circulant sur ce réseau. Afin d’examiner les paquets au contenu "remarquable", comme par exemple /etc/passwd, des signatures sont créées et nous y travaillerons tout au long de l’article. Principalement, des capteurs (souvent de simples hôtes) sont utilisés qui ne font rien d’autre que d’analyser le trafic et si nécessaire d’envoyer une alerte. Puisqu’il s’agit de leur unique tâche il est facile de mieux les sécuriser. Dans ce contexte, le "stealth mode" (mode furtif) est souvent choisi, c’est-à-dire que les capteurs agissent de manière invisible et il devient ainsi plus difficile pour un attaquant de les localiser et de les atteindre.

"Le mode furtif peut être utilisé pour les capteurs réseau afin de rendre invisible le mode "promiscuous" de la carte réseau puisque dans ce cas elle n’a pas d’adresse IP. On obtient ceci habituellement, en utilisant une carte distincte dans chaque machine du réseau servant de capteur, pour communiquer avec la console par un réseau isolé physiquement sécurisé. Le mode furtif complique la tâche du pirate qui voudrait attaquer le capteur réseau proprement dit."

Ce paragraphe (venant de la description de RealSecure) éclaircit la notion de capteur en mode furtif. Le capteur bascule en mode "promiscuous" (autrement dit, le mode dans lequel la carte réseau lit l’ensemble du trafic) et ne possède pas d’adresse IP propre. Ainsi, il est très difficile pour l’attaquant de localiser le capteur. D’ailleurs, ce mode est utilisé par les "renifleurs" de paquets comme tcpdump...

Vous pouvez mettre des capteurs dans les zones suivantes :

A l’intérieur du pare-feu (simplifié):



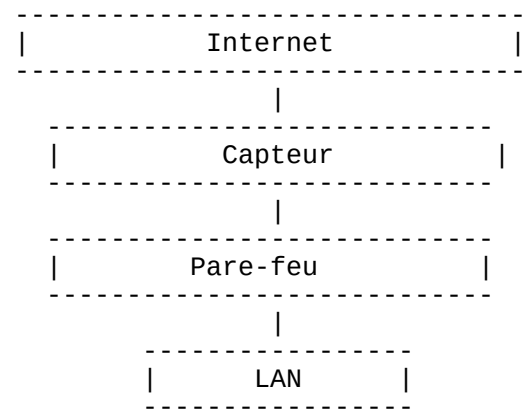
Il ne s’agit que d’une possibilité qui sert à montrer que les capteurs ne se situent pas entre la DMZ et le

pare-feu. Si l’expression DMZ ne vous dit rien, elle signifie "zone démilitarisée", autrement dit, une zone sécurisée de tous les côtés.

Si les capteurs se trouvent à l’intérieur du pare-feu, il leur est plus facile de dire si le pare-feu a été mal configuré et vous pouvez ainsi savoir si une attaque est venue par ce pare-feu. L’expérience a montré que les capteurs génèrent moins de faux positifs s’ils agissent "moins discrètement" tout en réduisant le trafic (d’où la diminution des fausses alertes).

Les capteurs placés à l’intérieur du pare-feu servent à la détection d’intrusion. Si votre capteur doit jouer ce rôle, alors mettez-le là...

A l’extérieur du pare-feu (simplifié):



Les capteurs sont souvent placés à l’extérieur du pare-feu externe comme indiqué sur le graphique. La raison en est que le capteur peut recevoir et analyser l’ensemble du trafic de l’Internet. Si vous placez le capteur ici, il n’est pas certain que toutes les attaques soient filtrées et détectées, par exemple les attaques TCP. Dans ce cas, vous devrez essayer de détecter l’attaque en utilisant des signatures (vous en saurez plus sur les signatures dans le chapitre correspondant). Pourtant, cet emplacement est le préféré de nombreux experts parce qu’il offre l’avantage d’écrire dans les logs et d’analyser les attaques (vers le pare-feu...), ainsi l’administrateur voit ce qu’il doit modifier dans la configuration du pare-feu.

Les capteurs placés à l’extérieur du pare-feu servent à la détection d’attaques (ce qui n’est pas le cas lorsqu’ils sont placés à l’intérieur du pare-feu !). Si vos capteurs doivent détecter ce type d’attaque, vous devrez donc les placer à l’extérieur du pare-feu...

- A l’intérieur et à l’extérieur du pare-feu

En fait, cette variante réunit les deux cas mentionnés ci-dessus. Mais elle est très dangereuse si vous configurez mal les capteurs et/ou le pare-feu, c’est-à-dire, vous ne pouvez pas simplement ajouter les avantages des deux cas précédents à cette variante. Ce ne sont pas, bien évidemment, les seuls endroits où placer des capteurs; vous pouvez en mettre dans bien d’autres lieux, mais ceux mentionnés ci-dessus sont les plus courants.

Avantages :

- les capteurs peuvent être bien sécurisés puisqu’ils se "contentent" d’observer le trafic

- vous détectez plus facilement les scans - grâce aux signatures... vous pouvez filtrer le trafic (nous verrons plus loin que ce n’est pas toujours le cas)

Inconvénients :

- la probabilité de faux négatifs (attaques non détectées comme telles) est élevée et il est difficile de contrôler le réseau entier
- ils doivent principalement fonctionner de manière cryptée d’où une complication de l’analyse des paquets
- à l’opposé des IDS basés sur l’hôte, ils ne voient pas les impacts d’une attaque

Exemples:

- NetRanger [http://www.cisco.com]
- Dragon [http://www.securitywizards.com]
- NFR [http://www.nfr.net]
- Snort [http://www.snort.org]
- DTK [http://all.net/dtk/dtk.html]
- ISS RealSecure [http://www.uh.edu/infotech/software/unix/realsecure/index.html]

Même si je distingue HIDS et NIDS, la différence devient de plus en plus réduite puisque les HIDS possèdent maintenant les fonctionnalités de base des NIDS. Des IDS bien connus comme ISS RealSecure se nomment aujourd’hui "IDS hôte et réseau". Dans un futur proche la différence entre les deux systèmes deviendra de plus en plus faible (ces sytèmes vont évoluer ensemble).

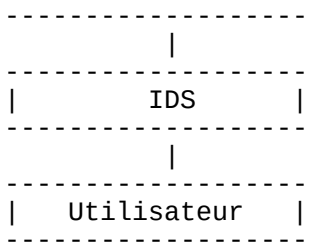
Système de Détection d’Intrusion de Noeud Réseau (NNIDS)

Ce nouveau type (NNIDS) fonctionne comme les NIDS classiques, c’est-à-dire vous analysez les paquets du trafic réseau. Mais ceci ne concerne que les paquets destinés à un noeud du réseau (d’où le nom). Une autre différence entre NNIDS et NIDS vient de ce que le NIDS fonctionne en mode "promiscuous", ce qui n’est pas le cas du NNIDS. Puisque tous les paquets ne sont pas analysés, les performances de l’ensemble sont améliorées.

Détection d’Intrusion basée sur une Application

Les IDS basés sur les applications sont un sous-groupe des IDS hôtes, mais je les mentionne séparément. Ils contrôlent l’interaction entre un utilisateur et un programme ajoutant des fichiers de log afin de fournir plus ample information sur les activités. Puisque vous opérez entre utilisateur et programme, il est facile de filtrer tout comportement notable. Un ABIDS peut être visualisé comme suit :

```
--------------------
|   Application    |
```



Avantages :

- vous travaillez en clair, contrairement aux NIDS par exemple, d'où une analyse plus facile
- vous pouvez détecter et empêcher des commandes particulières dont l'utilisateur pourrait se servir avec le programme

Inconvénients :

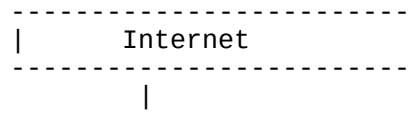
- faible sécurité et peu de possibilités de détecter, par exemple, un Cheval de Troie (puisque vous n'agissez pas dans l'espace du noyau)
- les fichiers de log générés par ce type d'IDS sont des cibles faciles pour les attaquants et ne sont pas aussi sûrs, par exemple, que les traces d'audit du système

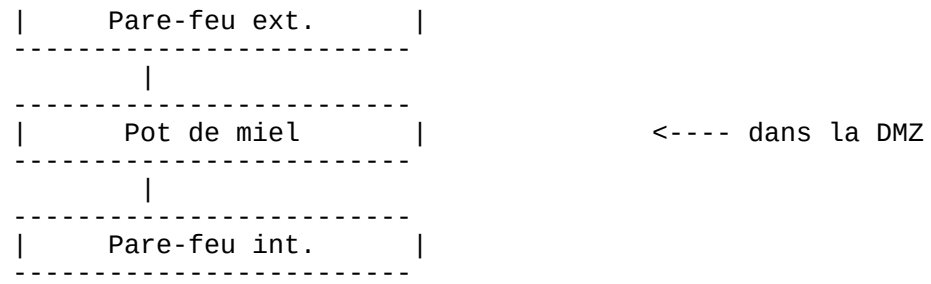
Détection d'Intrusion basée sur la Pile

Une autre nouvelle sorte d'IDS : Système de Détection d'Intrusion Basé sur la Pile (SBIDS). Mais pour l'instant, je n'ai pas suffisamment d'information pour vous en parler. Dans un article suivant, vous en trouverez certainement une description...

Honeypot (pot de miel)

Les "Pots de miel" possèdent de nombreuses similitudes avec les IDS, ils ont donc leur place ici. Voici une bonne définition trouvée dans la FAQ Intrusion Deception du SANS Institute : "Les Pots de miel sont des programmes qui simulent un ou plusieurs services réseau correspondants aux ports de votre ordinateur. Un attaquant pense que des services vulnérables sont actifs et qu'il peut les utiliser pour s'introduire dans votre machine. Un Pot de miel est utilisé pour créer des logs des tentatives d'accès à ces ports et aussi des frappes clavier de l'attaquant. Ceci permet d'obtenir des alertes précises dans le cas d'attaques élaborées". Dans certains cas, un pot de miel est une simple machine. De l'extérieur, elle apparaît vulnérable, alors qu'elle enregistre et analyse tout le trafic. Ainsi, comme ces pots de miel semblent vulnérables et qu'aucune connexion ne devrait avoir lieu, chaque tentative est perçue de manière suspecte.





Ainsi, le réseau interne n’est pas exposé puisque le pot de miel est généralement placé dans la DMZ (zone démilitarisée). La tâche principale d’un pot de miel consiste à analyser le trafic, c’est-à-dire, à informer du démarrage de certains processus, de la modification de fichiers... permettant ainsi de créer un profil élaboré des attaquants potentiels.

Mais tous les pots de miel ne sont pas semblables et il existe trois variantes qui améliorent la sécurité du système de manière différente :

Prévention : prévient d’une détection d’attaque; informe qu’une attaque a eu lieu (avec la localisation de l’attaquant, le type d’attaque, les services concernés...). Réaction : la détection par elle-même ne sert à rien si on ne prend pas certaines mesures de contre-attaque. Les "Padded Cell Systems" (systèmes "capitonnés") offrent de nombreuses possibilités de mesures réactives. Ainsi, la réponse dépend de ce que vous faites, de ce que fait l’IDS et de la manière dont vous réagissez à une attaque.

Nous reviendrons plus tard sur les différentes options de réponse des IDS... Enfin, les pots de miel peuvent être classés en deux catégories : recherche et production. Les pots de miel de production consistent à diminuer les risques sur un réseau alors que ceux de recherche servent à obtenir des informations sur les attaquants.

Prévention :
Les pots de miel ne sont pas les systèmes les plus appropriés pour éviter les attaques. Comme nous le verrons avec les "systèmes capitonnés", un pot de miel mal programmé et mal configuré peut même faciliter les attaques.

Détection:
De même, l’un des principaux avantages des pots de miel vient de ce qu’ils excellent à la détection d’attaques. Dans ce contexte, ils peuvent surtout analyser et interpréter les logs. Le placement du pot de miel joue un rôle décisif : les administrateurs ne peuvent bénéficier des pots de miel que s’ils sont placés et configurés correctement. Souvent, ils sont placés entre des serveurs importants pour détecter des scans éventuels du réseau entier ou à proximité d’un serveur essentiel pour détecter les accès illégaux à l’aide d’une redirection de port (c’est-à-dire, si quelqu’un essaie d’accéder au serveur par certains ports, il sera redirigé vers le pot de miel, et comme cet accès ne doit pas être autorisé, il y aura une alerte).

Réaction:
Dans le chapitre sur les Réponses (voir ci-dessous), vous découvrirez les possibilités que possède un pot de miel pour réagir aux événements. Si vous utilisez ou installez un pot de miel, gardez à l’esprit qu’un hôte est toujours très attractif pour un pirate, autrement dit cet hôte ne doit pas être trop difficile à attaquer. Il ne devrait pas y avoir trop de modifications par rapport à une configuration par défaut sinon cela se remarquerait. Pourtant, il n’est pas question d’oublier l’idée de départ : le contrôle du trafic, les

logs... Une approche dont j'ai souvent entendu parler consiste à placer le pot de miel dans son propre sous-réseau derrière un pare-feu. Ceci procure une bonne capacité d'alarme et permet donc d'afficher des alertes. Puisque les logs sont les cibles suivantes des attaquants, vous ne devez pas les gérer sur l'hôte lui-même mais les envoyer vers un autre serveur. Parfois, les "renifleurs" sont utilisés pour rechercher certains signes dans le trafic et les enregistrent. Si vous avez recueilli suffisamment d'information, analysez ces logs pour trouver les faiblesses du réseau et corrigez-les. En raison de la quantité d'information, il devrait être plus facile de boucher les trous de sécurité et de prendre des mesures contre les attaques les plus "méchantes". Mais vous devez savoir que les pots de miel ne sont pas totalement légaux et qu'il faut être prudent lorsqu'on les utilise. Le problème vient du fait qu'un pot de miel peut être considéré comme une "invitation à l'attaque" et si vous constatez que l'hôte a été attaqué (sans avoir pris de mesures réactives) ça peut aussi être interprété comme une grosse négligence. Quelques argumentations judiciaires contre les pots de miel semblent banales mais vérifiez quand même ce que dit la loi en vigueur dans votre pays. Si quelqu'un attaque un autre réseau à partir du votre et cause des dommages, vous serez considérés comme responsables pour les raisons mentionnées ci-dessus.
Des recherches approfondies ont montré que par exemple, en Europe, il n'y a aucun problème (si vous trouvez une loi en contradiction avec ce qui précède, merci de m'en informer; mes recherches n'ont rien trouvé de semblable).

Les Systèmes "Capitonnés" (Padded Cell Systems)

Ceux-ci fonctionnent habituellement avec l'un ou l'autre des systèmes abordés. Si l'IDS utilisé informe d'une attaque, l'attaquant est dirigé vers un hôte "capitonné". Une fois dans cet hôte, il ne peut plus causer de dommages puisque l'environnement est simulé, c'est un leurre. Cet environnement doit être aussi réaliste que possible, autrement dit, l'attaquant doit penser que son attaque a été couronnée de succès. Il est possible d'enregistrer, d'analyser et de suivre toute activité de l'attaquant. Le problème avec ces systèmes "capitonnés" vient de ce qu'ils peuvent être illégaux dans certains pays (de la même manière que certaines contre-mesures d'un IDS peuvent être considérées comme une attaque alors qu'elles ne servent qu'à obtenir des informations sur l'attaquant).

Avantages :

- une fois dans l'hôte "capitonné", les attaquants ne peuvent plus causer de dommages puisqu'il s'agit d'un environnement "bidon".
- l'administrateur peut suivre et enregistrer les activités de l'attaquant directement afin d'obtenir plus ample information sur l'attaque et sa cible, lui permettant ainsi de prendre des mesures réactives plus facilement

Inconvénients :

- l'usage de systèmes "capitonnés" peut être illégal (comme pour les pots de miel, il ne devrait pas y avoir de problème en Europe)
- la mise en oeuvre d'un tel système est assez difficile et réclame des compétences puisque l'ensemble de l'environnement doit être simulé correctement. Si l'administrateur fait une petite erreur, ce système peut alors être à l'origine de nouveaux trous de sécurité

Types d’attaques

Avant de développer ou d’utiliser un IDS, vous devez définir les dangers potentiels ainsi que le type d’attaque auxquels vous vous attendez. Malgré leur grande variété, les attaques et leurs cibles peuvent se ranger en quatre catégories :

1)Confidentialité
Les conséquences de l’attaque sont que la confiance mutuelle envers un utilisateur a changé (en général en sa faveur), c’est-à-dire qu’il n’a plus à s’authentifier pour un programme donné. De telles circonstances sont encore très répandues de nos jours : si par exemple, une confiance mutuelle est instaurée entre deux hôtes, l’utilisateur d’un hôte peut se loger dans un autre hôte sans mot de passe. Si un attaquant parvient à un équivalent de cette situation par son attaque, il sera souvent très difficile de découvrir cet "abus de confiance".

2)Intégrité
Si l’utilisateur modifie d’importants fichiers système ou de configuration, remplace des binaires par les siens... Souvent, des binaires (comme par exemple /bin/login) sont remplacés par des versions "personnelles". L’utilisateur doit alors fournir un mot de passe défini (dans le code source) et obtient (principalement) des privilèges de root. De telles attaques servent à augmenter les privilèges d’un utilisateur, par exemple, en installant un binaire login modifié (backdoor). Il existe des outils tels que Tripwire pour lutter contre ces "backdoors" mais tout le monde ne les utilise pas. De plus, des erreurs dans la configuration et l’utilisation sont souvent dévastatrices et font de Tripwire un risque supplémentaire dans ces cas-là.

3)Disponibilité
L’accessibilité du système est affectée. Ceci peut empêcher certains utilisateurs de se connecter de manière définitive ou occasionnelle. Le but est de travailler sans être dérangé et de ne pas être découvert.

4)Contrôle
Par exemple, si l’attaque est destinée à prendre le contrôle du système, c’est-à-dire les fichiers, les programmes... Si cela se produit, vous pouvez considérer que l’attaquant bénéficie également des trois autres possibilités. S’il prend le contrôle total sur le système, il peut modifier, restreindre tout ce qu’il veut.

Avant d’en arriver aux différents types d’attaques (Dos, DDoS, Scans...), je vais faire une petite incursion dans le monde des contrôleurs d’intégrité. Des outils comme Tripwire font partie des IDS basés sur l’hôte, et le but d’un contrôleur d’intégrité est de vérifier l’intégrité de différents fichiers et de déclencher une alarme s’il détecte une modification. La procédure d’utilisation des contrôleurs d’intégrité est toujours équivalente, il n’est donc pas nécessaire de décrire un contrôleur spécifique.

1) Création d’une base de données
La première étape suivant l’installation d’un contrôleur d’intégrité comme Tripwire consiste à créer une base de données. Cette étape devrait (doit) être exécutée dans la situation d’un système non compromis. Si vous créez la base ultérieurement (et qu’un attaquant a déjà remplacé certains binaires) le contrôleur d’intégrité ne fonctionnera plus comme il le devrait. Dans un tel cas, les binaires remplacés sont

considérés comme des originaux et si l'administrateur les remplace par les véritables originaux, une alarme sera déclenchée. La plupart des outils offrent des possibilités étendues pour spécifier des fichiers ou des répertoires pour lesquels vous voulez créer des "checksums". Autrement dit vous générez une empreinte du système. (Tripwire appelle cela le Mode d'Initialisation de la Base de données)

2) Vérification d'intégrité
Dès que l'administrateur possède sa base de données (l'empreinte) du système, il peut vérifier ce système quand il le souhaite. Grâce à la base créée, il redéfinit les "checksums", les compare à ceux de la base de données et si une valeur diffère, une alarme est déclenchée. Tripwire offre encore plus de possibilités. Lisez sa page de manuel ou la documentation associée.

Ces deux étapes sont communes à la plupart des contrôleurs d'intégrité. Tripwire offre la possibilité de mettre la base à jour (après avoir installé de nouveaux outils) ou de changer la politique d'utilisation, ou de tester le système d'avertissement par courrier.

Quelles sont les erreurs possibles lors de l'utilisation de contrôleurs d'intégrité ?

La première et plus grossière erreur qu'un administrateur puisse faire est de créer un base de checksums MD5 lorsqu'il existe une forte probabilité que des binaires aient été remplacés. Si un administrateur suppose qu'une attaque a pu aboutir et crée la base de ce système compromis a posteriori, le binaire remplacé par l'attaquant (par exemple un "login backdoor"), sera considéré comme le véritable binaire. Vous devez créer la base de données immédiatement après installation. Une autre erreur grossière dans l'utilisation des contrôleurs d'intégrité (comme Tripwire, par exemple) consiste à laisser la base sur le disque dur. A première vue, il n'y a rien de choquant à laisser la base sur le disque, mais si vous devez le faire, vérifiez que la partition soit en lecture seule. Vous devez vérifier que personne ne possède un accès en écriture à votre base. Si un attaquant peut écrire dans votre base, il pourra modifier tout ce qu'il veut. Si vous avez déjà travaillé avec Tripwire, vous savez certainement que vous pouvez adapter le fichier de configuration aux fichiers dont l'intégrité doit être contrôlée. Mais que se passe-t-il si l'attaquant peut lire et écrire dans ce fichier de configuration ? Il peut modifier les chemins de recherche de manière à ce que le répertoire contenant les binaires modifiés ne soit pas scanné. Il est préférable de ne pas laisser le fichier de configuration sur le disque dur et de l'enregistrer sur un support en lecture seule. Certains penseront que c'est du temps perdu d'enregistrer les fichiers sur un support en lecture seule, pourtant il est préférable de passer un peu plus de temps afin d'améliorer la sécurité (un aspect intéressant de la convivialité vient de ce qu'elle a ouvert de nombreux trous de sécurité. Dans notre cas nous ne recherchons pas la convivialité mais un meilleur niveau de sécurité). Tripwire (et d'autres contrôleurs d'intégrité) sont certainement des programmes capables d'améliorer la sécurité et de rendre la tâche difficile aux attaquants potentiels. Mais le meilleur outil devient inutile si les autres paramètres de sécurité du système sont impropres. Par conséquent, ne demandez pas aux contrôleurs d'intégrité de faire tout votre travail, faites vous-mêmes ce qui doit l'être sur votre hôte.

Un nouveau type de contrôleurs d'intégrité se nomme contrôleurs d'intégrité en temps réel. Contrairement aux contrôleurs classiques, ils vérifient l'intégrité en temps réel. Voici un bref aperçu :

1) Ici aussi, la première étape est la création de la base de données des checksums.
2) Avant que quiconque puisse exécuter un programme, le checksum du fichier est généré. Si sa valeur ne correspond pas à celle de la base, le binaire a été remplacé. Si c'est le cas, l'exécution est interdite. Ce concept est le seul possible pour les contrôleurs d'intégrité en temps réel (tout au moins, c'est ainsi que je le conçois). Contrairement aux contrôleurs habituels, vous ne dépendez pas de la vérification

régulière des fichiers par l'administrateur, vous essayez de vérifier la conformité du binaire avant exécution et si nécessaire vous ne le laissez pas s'exécuter.

En dehors des catégories mentionnées ci-dessus (intégrité, contrôle...), les attaques peuvent être différenciées autrement, par exemple comment attaquer, par quels moyens. Certes, il existe de nombreuses possibilités, mais les attaques les plus communes contre les IDS sont les suivantes :

- Les Scans

Aujourd'hui, les scans sont devenus des attaques courantes, le problème étant que l'IDS ne doit pas générer trop de faux positifs. Les scans servent principalement à obtenir de l'information sur un hôte, un réseau (pour préparer une attaque plus élaborée). Les informations qu'un attaquant peut obtenir sur votre réseau figurent dans le résultat du scan ci-dessous (nmap - un simple exemple qui ne montre pas toutes les possibilités de nmap) :

```
....
Host (192.168.0.0) seems to be a subnet broadcast address (returned 1
extra pings).
Skipping host. Interesting ports on playground.yuma.net 192.168.0.1):
Port           State          Protocol               Service
22             open           tcp            ssh
111            open           tcp            sunrpc
635            open           tcp            unknown
1024           open           tcp            unknown
2049           open           tcp            nfs

TCP Sequence Prediction: Class = random positive increments
                         Difficulty=3916950 (Good luck!)
Remote operating system guess:
       Linux 2.1.122 - 2.1.132; 2.2.0-pre1 - 2.2.2

Interesting ports on vectra.yuma.net (192.168.0.5):
Port           State          Protocol               Service
13             open           tcp            daytime
21             open           tcp            ftp
22             open           tcp            ssh
23             open           tcp            telnet
37             open           tcp            time
79             open           tcp            finger
111            open           tcp            sunrpc
113            open           tcp            auth
513            open           tcp            login
514            open           tcp            shell

TCP Sequence Prediction: Class = random positive increments
                         Difficulty = 17719 (Worthy challenge)
Remote operating system guess: OpenBSD 2.2. - 2.3

Nmap run completed -- 256 IP addresses (2 hosts up) scanned in 6 seconds
```

Principalement, vous obtenez ce qui suit :

- le système d'exploitation de la machine
- les versions de certains programmes actifs
- les services actifs qu'il est possible d'attaquer
- les ports ouverts
- ...

La plupart des techniques de scan offrent une énorme quantité d'information qui permet à l'attaquant de préparer son méfait. Je ne vais certes pas décrire ou mentionner toutes les techniques de scan, mais je citerai les plus utilisées (si vous avez des questions sur les protocoles, référez-vous aux RFC correspondantes) :

Ping scanning:
Ce type de scan est utilisé pour découvrir quels sont les hôtes connectés. Pour ce faire, vous envoyez à l'hôte (ou aux hôtes) un datagramme ICMP de type 8 (echo request) et attendez une réponse contenant un datagramme ICMP de type 0 (echo reply). Parfois, vous n'envoyez pas seulement un "echo request" mais aussi un ACK puisque ICMP est parfois bloqué. Si une réponse RST est reçue, l'hôte est connecté.

Paramètre Nmap : -sP

TCP Scanning (Vanilla) :
Avec le "TCP scanning" vous essayez de vous connecter sur tous les ports après le three-way-handshake, par exemple, vous avez établi avec succès une connexion sur les ports et la valeur de réponse de connect() est vérifiée. Pour l'attaquant, la valeur de réponse indique si le(s) port(s) utilisé(s) est ouvert ou fermé. Le but d'un scan TCP est de savoir si les ports sont ouverts ou fermés.

Paramètre Nmap : -sT

La sortie suivante de nmap est celle de l'un de mes hôtes, immédiatement après une installation par défaut. Je n'ai volontairement fait aucun changement (dans la configuration) :

```
 [Socma]$ nmap -sT localhost

Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Interesting ports on Diablo (127.0.0.1):
(The 1552 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
23/tcp     open        telnet
80/tcp     open        http
111/tcp    open        sunrpc
113/tcp    open        auth
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second
```

Un extrait de la trace de tcpdump (pour ce scan):

```
 02:10:15.804704 Diablo > Diablo: icmp: echo request
                          4500 001c 2db8 0000 3501 5a27 7f00 0001
                          7f00 0001 0800 fc95 fb69 0000
 02:10:15.814704 Diablo > Diablo: icmp: echo reply (DF)
                          4500 001c 0000 4000 ff01 7dde 7f00 0001
                          7f00 0001 0000 0496 fb69 0000
 02:10:15.814704 Diablo.58725 > Diablo.http: . ack 110306597 win 3072
                          4500 0028 d223 0000 2a06 c0aa 7f00 0001
                          7f00 0001 e565 0050 ad48 0003 0693 2525
                          5010 0c00 e718 0000
 02:10:15.814704 Diablo.http > Diablo.58725: R 110306597:110306597(0)
```

```
        win 0 (DF)
                        4500 0028 0000 4000 ff06 7dcd 7f00 0001
                        7f00 0001 0050 e565 0693 2525 0000 0000
                        5004 0000 a070 0000
02:10:16.114704 Diablo.1727 > Diablo.821: S 196002918:196002918(0)
win 32767 <mss 16396,sackOK,timestamp 213509[|tcp]> (DF)
                        4500 003c 8663 4000 4006 b656 7f00 0001
                        7f00 0001 06bf 0335 0bae c466 0000 0000
                        a002 7fff 739c 0000 0204 400c 0402 080a
                        0003 4205
02:10:16.114704 Diablo.821 > Diablo.1727: R 0:0(0) ack 196002919
       win 0 (DF)
                        4500 0028 0000 4000 ff06 7dcd 7f00 0001
                        7f00 0001 0335 06bf 0000 0000 0bae c467
                        5014 0000 d7c4 0000
02:10:16.114704 Diablo.1728 > Diablo.440: S 195504823:195504823(0)
win 32767 <mss 16396,sackOK,timestamp 213509[|tcp]> (DF)
                        4500 003c 68b2 4000 4006 d407 7f00 0001
                        7f00 0001 06c0 01b8 0ba7 2ab7 0000 0000
                        a002 7fff 0ecf 0000 0204 400c 0402 080a
                        0003 4205
```

UDP scanning:
Le but des scans UDP est équivalent à celui des scans TCP, puisqu'il s'agit de découvrir les ports UDP ouverts. Le scan travaille différemment puisqu'UDP est un protocole sans connexion (TCP est un protocole orienté connexion). Un scan UDP utilise ICMP, si vous envoyez au port concerné un paquet UDP de 0 octet et attendez une réponse ICMP. Si vous recevez une information comme quoi le port est inaccessible ("Port unreachable"/code value 3), cela signifie que le port est fermé. Si l'administrateur a désactivé certains services dans /etc/inetd.conf et que vous essayiez d'envoyer un paquet vers le port concerné, vous obtiendrez le message d'erreur "Port Unreachable".

Paramètre Nmap : -sU

```
 [Socma]$ nmap -sU localhost

Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Interesting ports on Diablo (127.0.0.1):
(The 1459 ports scanned but not shown below are in state: closed)
Port       State       Service
111/udp    open        sunrpc

Nmap run completed -- 1 IP address (1 host up) scanned in 4 seconds
```

La trace tcpdump associée :

```
10:41:55.954397 Diablo > Diablo: icmp: echo request
                        4500 001c cc8f 0000 2801 c84f 7f00 0001
                        7f00 0001 0800 8471 738e 0000
10:41:55.954397 Diablo > Diablo: icmp: echo reply (DF)
                        4500 001c 0000 4000 ff01 7dde 7f00 0001
                        7f00 0001 0000 8c71 738e 0000
10:41:55.964397 Diablo.63793 > Diablo.http: . ack 994287972 win 2048
                        4500 0028 79e3 0000 2506 1deb 7f00 0001
                        7f00 0001 f931 0050 06d8 0003 3b43 a164
                        5010 0800 cccd 0000
```

```
10:41:55.964397 Diablo.http > Diablo.63793: R 994287972:994287972(0)
     win 0 (DF)
                          4500 0028 0000 4000 ff06 7dcd 7f00 0001
                          7f00 0001 0050 f931 3b43 a164 0000 0000
                          5004 0000 dbb4 0000
10:41:56.274397 Diablo.63773 > Diablo.15: udp 0
                          4500 001c 8a0b 0000 3011 02c4 7f00 0001
                          7f00 0001 f91d 000f 0008 08af
10:41:56.274397 Diablo > Diablo: icmp: Diablo udp port 15
     unreachable (DF) [tos 0xc0]
                          45c0 0038 0000 4000 ff01 7d02 7f00 0001
                          7f00 0001 0303 fb18 0000 0000 4500 001c
                          8a0b 0000 3011 02c4 7f00 0001 7f00 0001
                          f91d 000f
10:41:56.274397 Diablo.63773 > Diablo.1459: udp 0
                          4500 001c 6c2f 0000 3011 20a0 7f00 0001
                          7f00 0001 f91d 05b3 0008 030b
10:41:56.274397 Diablo > Diablo: icmp: Diablo udp port 1459
     unreachable (DF) [tos 0xc0]
                          45c0 0038 0100 4000 ff01 7c02 7f00 0001
                          7f00 0001 0303 fb18 0000 0000 4500 001c
                          6c2f 0000 3011 20a0 7f00 0001 7f00 0001
                          f91d 05b3
```

Une autre variante de scan UDP (scan UDPrecvfrom() et write()) consiste à scanner chaque port deux fois. La méthode utilise ICMP et "Port Unreachable", mais seul root reçoit ce message. Si vous scannez un port fermé deux fois, vous obtenez après le second scan : "Error 13 : Try Again"...

ACK scanning:
En envoyant un paquet ACK vers le port d'un pare-feu, vous découvrez les ports filtrés et non filtrés. Si vous recevez une réponse RST, elle signifie que le port correspondant n'est pas surveillé, par conséquent non filtré, sinon, vous obtenez un message d'erreur ICMP. Vous ne découvrez pas les ports ouverts mais vous avez une information plus précise sur le pare-feu (et sa configuration).

Paramètre Nmap : -sA

```
 [Socma]$ nmap -sA localhost

Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
All 1558 scanned ports on Diablo (127.0.0.1) are: UNfiltered

Nmap run completed -- 1 IP address (1 host up) scanned in 6 seconds

 The tcpdump trace:

10:45:51.864397 Diablo > Diablo: icmp: echo request
                          4500 001c 1617 0000 3901 6dc8 7f00 0001
                          7f00 0001 0800 113d e6c2 0000
10:45:51.864397 Diablo > Diablo: icmp: echo reply (DF)
                          4500 001c 0000 4000 ff01 7dde 7f00 0001
                          7f00 0001 0000 193d e6c2 0000
10:45:51.864397 Diablo.53119 > Diablo.http: . ack 2682022466 win 3072
                          4500 0028 0dda 0000 3206 7cf4 7f00 0001
                          7f00 0001 cf7f 0050 0650 0003 9fdc 6a42
                          5010 0c00 c590 0000
10:45:51.864397 Diablo.http > Diablo.53119: R 2682022466:2682022466(0)
     win 0 (DF)
                          4500 0028 0000 4000 ff06 7dcd 7f00 0001
```

```
                          7f00 0001 0050 cf7f 9fdc 6a42 0000 0000
                          5004 0000 d7ef 0000
10:45:52.164397 Diablo.53099 > Diablo.14: . ack 2457451592 win 3072
                          4500 0028 218d 0000 3206 6941 7f00 0001
                          7f00 0001 cf6b 000e 5938 4710 9279 bc48
                          5010 0c00 e74d 0000
10:45:52.164397 Diablo.14 > Diablo.53099: R 2457451592:2457451592(0)
     win 0 (DF)
                          4500 0028 0000 4000 ff06 7dcd 7f00 0001
                          7f00 0001 000e cf6b 9279 bc48 0000 0000
                          5004 0000 93a2 0000
10:45:52.164397 Diablo.53099 > Diablo.imap3: . ack 2457451592 win 3072
                          4500 0028 a075 0000 3206 ea58 7f00 0001
                          7f00 0001 cf6b 00dc 5938 4710 9279 bc48
                          5010 0c00 e67f 0000
```

Stealth scanning (NULL, XMAS, FIN, SYN, ...): Avec le "stealth scanning" vous "abusez" le three-way-handshake. Il serait préférable d'expliquer comment il fonctionne puisqu'il permet d'expliquer les scans stealth, mais je présente brièvement un three-way-handshake :

```
------------------      SYN                         -------------
| Host    A      | ---------------------------- > | Host    B |
------------------                                  -------------

------------------              ACK/SYN             -------------
| Host    A      | <----------------------------- | Host    B |
------------------                                  -------------

------------------              ACK                 -------------
| Host    A      | ---------------------------- > | Host    B |
------------------                                  -------------
```

Le problème des scans TCP c'est qu'ils se remarquent (puisqu'il y a un three-way-handshake à chaque fois). Voici au contraire, ce qui se produit avec un scan stealth (furtif):

```
------------------      SYN                         -------------
| Host    A      | ---------------------------- > | Host    B |
------------------                                  -------------

------------------              ACK/SYN             -------------
| Host    A      | <----------------------------- | Host    B |
------------------                                  -------------
```

Ce diagramme ressemble au three-way-handshake mais avec une différence : il n'y a pas de connexion entre A et B. L'hôte B pense que la connexion existe alors qu'elle n'existe pas tant que A n'envoie pas un ACK supplémentaire à B (on appelle cela un port à moitié ouvert). Le scan SYN montré ci-dessus implique l'ouverture du port sur l'hôte cible (en raison du ACK/SYN), s'il était fermé on obtiendrait en retour un RST/ACK.

Paramètre Nmap : -sS

```
 [Socma]$ nmap -sS localhost

Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Interesting ports on Diablo (127.0.0.1):
```

```
(The 1552 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
23/tcp     open        telnet
80/tcp     open        http
111/tcp    open        sunrpc
113/tcp    open        auth
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 3 seconds

Trace tcpdump :

10:47:41.674397 Diablo > Diablo: icmp: echo request
                         4500 001c 8f08 0000 3501 f8d6 7f00 0001
                         7f00 0001 0800 99a9 5e56 0000
10:47:41.674397 Diablo > Diablo: icmp: echo reply (DF)
                         4500 001c 0000 4000 ff01 7dde 7f00 0001
                         7f00 0001 0000 a1a9 5e56 0000
10:47:41.674397 Diablo.58038 > Diablo.http: . ack 1561498752 win 3072
                         4500 0028 afe5 0000 3206 dae8 7f00 0001
                         7f00 0001 e2b6 0050 82b0 0003 5d12 9480
                         5010 0c00 4e85 0000
10:47:41.674397 Diablo.http > Diablo.58038: R 1561498752:1561498752(0)
     win 0 (DF)
                         4500 0028 0000 4000 ff06 7dcd 7f00 0001
                         7f00 0001 0050 e2b6 5d12 9480 0000 0000
                         5004 0000 dd44 0000
10:47:41.984397 Diablo.58018 > Diablo.1488: S 2803535203:2803535203(0)
     win 3072
                         4500 0028 a4f5 0000 3206 e5d8 7f00 0001
                         7f00 0001 e2a2 05d0 a71a 8d63 0000 0000
                         5002 0c00 88ef 0000
10:47:41.984397 Diablo.1488 > Diablo.58018: R 0:0(0) ack 2803535204
     win 0 (DF)
                         4500 0028 0000 4000 ff06 7dcd 7f00 0001
                         7f00 0001 05d0 e2a2 0000 0000 a71a 8d64
                         5014 0000 94dc 0000
```

Voici maintenant d'autres types de scans : le FIN, le NULL et le XMAS. Le scan FIN envoie seulement un message FIN à l'hôte cible, même s'il n'existe pas de connexion entre eux. Si le port est fermé un RST est renvoyé, sinon, rien ne se passe.

Paramètre Nmap : -sF

```
 [Socma]$ nmap -sF localhost

Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Interesting ports on Diablo (127.0.0.1):
(The 1552 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
23/tcp     open        telnet
80/tcp     open        http
111/tcp    open        sunrpc
113/tcp    open        auth
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 6 seconds
```

```
Trace tcpdump :

10:48:28.704397 Diablo > Diablo: icmp: echo request
                        4500 001c b29d 0000 3401 d641 7f00 0001
                        7f00 0001 0800 a1a7 5658 0000
10:48:28.704397 Diablo > Diablo: icmp: echo reply (DF)
                        4500 001c 0000 4000 ff01 7dde 7f00 0001
                        7f00 0001 0000 a9a7 5658 0000
10:48:28.704397 Diablo.52201 > Diablo.http: . ack 2873378189 win 4096
                        4500 0028 cbeb 0000 2b06 c5e2 7f00 0001
                        7f00 0001 cbe9 0050 9020 0003 ab44 458d
                        5010 1000 54a3 0000
10:48:28.704397 Diablo.http > Diablo.52201: R 2873378189:2873378189(0)
     win 0 (DF)
                        4500 0028 0000 4000 ff06 7dcd 7f00 0001
                        7f00 0001 0050 cbe9 ab44 458d 0000 0000
                        5004 0000 f4d2 0000
10:48:29.004397 Diablo.52181 > Diablo.233: F 0:0(0) win 4096
                        4500 0028 10c6 0000 2b06 8108 7f00 0001
                        7f00 0001 cbd5 00e9 0000 0000 0000 0000
                        5001 1000 d522 0000
10:48:29.004397 Diablo.233 > Diablo.52181: R 0:0(0) ack 1 win 0 (DF)
                        4500 0028 0000 4000 ff06 7dcd 7f00 0001
                        7f00 0001 00e9 cbd5 0000 0000 0000 0001
                        5014 0000 e50e 0000
```

Les scans NULL et XMAS offrent un intérêt particulier (surtout utilisés conjoitement à une détection d'anomalie de protocole). Il se nomme XMAS parce que tous les drapeaux sont actifs : SYN, ACK, FIN, URG, PUSH. Comme pour les scans FIN, un RST est renvoyé si le port est fermé.

Paramètre Nmap : -sX

```
 [Socma]$ nmap -sX localhost

Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Interesting ports on Diablo (127.0.0.1):
(The 1552 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
23/tcp     open        telnet
80/tcp     open        http
111/tcp    open        sunrpc
113/tcp    open        auth
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 5 seconds

Trace tcpdump :

10:44:24.004397 Diablo > Diablo: icmp: echo request
                        4500 001c ffcc 0000 2a01 9312 7f00 0001
                        7f00 0001 0800 103d e7c2 0000
10:44:24.004397 Diablo > Diablo: icmp: echo reply (DF)
                        4500 001c 0000 4000 ff01 7dde 7f00 0001
                        7f00 0001 0000 183d e7c2 0000
10:44:24.004397 Diablo.36398 > Diablo.http: . ack 718216305 win 2048
                        4500 0028 2e28 0000 2906 65a6 7f00 0001
                        7f00 0001 8e2e 0050 9220 0003 2acf 1c71
                        5010 0800 41f0 0000
10:44:24.004397 Diablo.http > Diablo.36398: R 718216305:718216305(0)
```

```
     win 0 (DF)
                          4500 0028 0000 4000 ff06 7dcd 7f00 0001
                          7f00 0001 0050 8e2e 2acf 1c71 0000 0000
                          5004 0000 dc1f 0000
10:44:24.304397 Diablo.36378 > Diablo.1996: FP 0:0(0) win 2048 urg 0
                          4500 0028 7651 0000 2906 1d7d 7f00 0001
                          7f00 0001 8e1a 07cc 0000 0000 0000 0000
                          5029 0800 13d3 0000
10:44:24.304397 Diablo.1996 > Diablo.36378: R 0:0(0) ack 1 win 0 (DF)
                          4500 0028 0000 4000 ff06 7dcd 7f00 0001
                          7f00 0001 07cc 8e1a 0000 0000 0000 0001
                          5014 0000 1be7 0000
```

L’autre possibilité, nommée scan NULL, signifie qu’aucun drapeau n’est actif, et si le port est fermé un RST est renvoyé.

Paramètre Nmap : -sN

```
 [Socma]$ nmap -sN localhost

Starting nmap V. 2.54BETA36 ( www.insecure.org/nmap/ )
Interesting ports on Diablo (127.0.0.1):
(The 1552 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
23/tcp     open        telnet
80/tcp     open        http
111/tcp    open        sunrpc
113/tcp    open        auth
6000/tcp   open        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 5 seconds

Trace tcpdump :

10:43:37.594397 Diablo > Diablo: icmp: echo request
                          4500 001c 2ecf 0000 2c01 6210 7f00 0001
                          7f00 0001 0800 8f87 6878 0000
10:43:37.594397 Diablo > Diablo: icmp: echo reply (DF)
                          4500 001c 0000 4000 ff01 7dde 7f00 0001
                          7f00 0001 0000 9787 6878 0000
10:43:37.604397 Diablo.34607 > Diablo.http: . ack 1932747046 win 4096
                          4500 0028 ee0f 0000 3706 97be 7f00 0001
                          7f00 0001 872f 0050 5b20 0003 7333 6126
                          5010 1000 ead5 0000
10:43:37.604397 Diablo.http > Diablo.34607: R 1932747046:1932747046(0)
     win 0 (DF)
                          4500 0028 0000 4000 ff06 7dcd 7f00 0001
                          7f00 0001 0050 872f 7333 6126 0000 0000
                          5004 0000 5605 0000
10:43:37.904397 Diablo.34587 > Diablo.408: . win 4096
                          4500 0028 e3bb 0000 3706 a212 7f00 0001
                          7f00 0001 871b 0198 0000 0000 0000 0000
                          5000 1000 192f 0000
10:43:37.904397 Diablo.408 > Diablo.34587: R 0:0(0) ack 0 win 0 (DF)
                          4500 0028 0000 4000 ff06 7dcd 7f00 0001
                          7f00 0001 0198 871b 0000 0000 0000 0000
                          5014 0000 291b 0000
```

Puisque le "three-way-handshake" complet n’est pas nécessaire, le scan furtif est plus discret que le scan

TCP (comme déjà mentionné). Un IDS devrait toujours détecter ce genre d'anomalies (XMAS et NULL)...

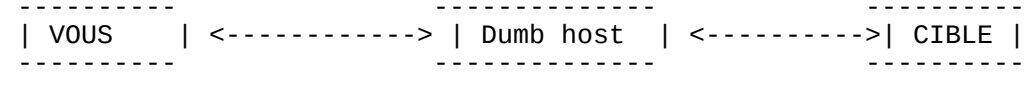
FTP bounce:
Avec certains ftpd, la commande PORT peut être abusée afin d'établir une connexion depuis le serveur ftp vers une autre machine. Mais d'abord, voici un aperçu de ce qui se passe "normalement". Le client établit une connexion avec le serveur ftp (port 21), le serveur ftp crée une seconde connexion vers le client (pour pouvoir lui renvoyer des données). Cette seconde connexion utilise la commande PORT. Le point intéressant ici, c'est que la commande contient l'adresse IP et le port (qui doit être ouvert) du client. Le serveur crée donc une connexion dans laquelle le port source 20 et le port de destination sont ceux spécifiés par la commande PORT. Le point d'attaque est la commande PORT avec laquelle vous pouvez manipuler le port du client (supposé) pour vous connecter à la victime au lieu de l'hôte réel. Après avoir manipulé l'adresse IP et le port, vous pouvez initialiser le trafic réel par "list" ou "get". Maintenant vous devez vérifier la réponse de ftp : si vous obtenez "425: Can't build data connection: Connection refused", le port spécifié est fermé. Si vous recevez : "150 : File status okay about to open data connection" ou "226: Closing data connection. Requested file action successful" comme réponse, alors le port est ouvert.

Paramètre Nmap : -b

Paquets fragmentés : cette méthode utilise la fragmentation par TCP d'un paquet IP. Habituellement, la fragmentation se produit lorsque les datagrammes sont plus gros que la taille permise; cette limitation se nomme MTU (Maximum Transmission Unit). Les datagrammes fragmentés sont reconstitués à la fin d'un noeud. Ce comportement peut être abusé. Tous les IDS ou pare-feu ne sont pas capables de gérer la fragmentation, générant parfois des erreurs avec les paquets fragmentés. Au lieu d'envoyer notre paquet normalement, nous le découpons en fragments. Ils contiennent des données communes comme l'adresse IP source, l'adresse IP de destination, le port source... Maintenant, il se peut que le pare-feu ou l'IDS aient du mal à reconstituer les fragments. Ces problèmes peuvent se présenter de différentes manières, soit en aboutissant à un "crash" du système, soit en laissant passer les paquets. Notre paquet peut passer puisque sa reconstitution est erronée et qu'il a été faussement défini comme inoffensif. Parfois, tous les fragments ne sont pas correctement vérifiés, autrement dit seule une partie est contrôlée, laissant ainsi passer le paquet. Avec cette technique vous pouvez scanner plus discrètement puisque le trafic est déclaré inoffensif et ne déclenche pas d'alerte. D'un autre côté, cette théorie dépend de la capacité de l'IDS ou du pare-feu à analyser et à reconstituer les fragments.

Scan Reverse Ident :
Un chapitre de la RFC 1413, celui concernant le protocole d'identification ("Identification Protocol ou "ident") propose un moyen de déterminer l'identité de l'utilisateur d'une connexion TCP spécifique. Selon un couple de numéros de ports TCP, il renvoie une chaîne de caractères identifiant le propriétaire de cette connexion vers le serveur. Le scan Reverse Ident utilise identd pour connaître le propriétaire du processus actif. Cette technique est particulièrement destinée à découvrir les démons fonctionnant en tant que root afin de les attaquer.

Scan Dumb :



Un Dumb host doit avoir un trafic aussi réduit que possible. Pourquoi est-il nécessaire d'utiliser un dumb host ? Ces questions mènent à la véritable attaque et aussi à l'explication de la nécessité d'un dumb host. Afin de découvrir si un port de "CIBLE" est ouvert ou fermé, vous devez examiner le champ ID de l'adresse IP du dumb host. Pour ce faire, vous devez lui envoyer un paquet (echo request) et grâce à sa réponse vous pouvez lire le champ ID (sa valeur). Vous pouvez donc envoyer un paquet à CIBLE dans lequel l'adresse source est celle du dumb host. La réponse est donc reçue par ce dumb host. S'il reçoit un SYN/ACK de CIBLE, ça signifie que le port est ouvert. Pour répondre, notre dumb host enverra un paquet dans lequel le drapeau RST est activé. S'il reçoit un RST/ACK de CIBLE, il ne répondra pas. Pour connaître les réponses que le dumb host a reçu de CIBLE, il faut envoyer un autre ping à dumb host. Si le port de la cible est ouvert et qu'il a répondu par RST, le champ ID de l'adresse IP du dumb host sera incrémenté. Si le port est fermé, rien ne se passe. En lisant la nouvelle valeur d'ID de l'adresse IP du dumb host, vous pouvez savoir si les ports sont ouverts ou fermés. La raison pour laquelle nous utilisons un dumb host (un hôte avec peu de trafic) devrait maintenant être plus claire. Si le trafic est trop important il sera beaucoup plus difficile de découvrir les ports ouverts (sur CIBLE), et la probabilité de commettre une erreur sera proportionnellement plus élevée.

Détection de l'empreinte de l'OS :
La détection de l'empreinte du système d'exploitation a pour but de découvrir le type d'OS du serveur. La plupart des nouveaux scanners ne fournissent plus seulement le nom comme "Linux" ou "Solaris" mais aussi la version correspondante. Pour ce faire, vous prenez une "empreinte" du système d'exploitation. Vous ne pouvez plus avoir confiance dans les bannières de telnet, ftp, puisqu'elles peuvent être manipulées (voir ci-dessus). Si l'attaquant découvre la version exacte du système de l'hôte cible, il peut créer des scripts, des exploits... et améliorer son attaque. Cette technique exploite les différences (souvent les plus minimes) entre les systèmes. Comme il existe une grande quantité de documentation sur cette technique je ne ferai qu'un bref tour d'horizon des possibilités de test les plus importantes :

- test FIN : si un hôte reçoit un paquet FIN sur un port ouvert, il ne doit pas répondre (RFC 793) mais il y a des exceptions, par exemple avec MS Windows, BSDI, CISCO, MPS, ... qui envoient un RESET comme réponse.

- les numéros séquentiels ACK : en envoyant FIN|PSH|URG vers un port fermé le numéro séquentiel du paquet ACK correspond à son propre numéro. Mais, ici aussi Windows fait exception, en envoyant son propre numéro séquentiel +1

- test du drapeau BOGUS : si un drapeau TCP indéfini est utilisé dans l'entête TCP (dans un paquet SYN), les hôtes Linux < 2.0.35 adoptent ce drapeau. D'autres OS redémarrent à la réception d'un paquet SYN+BOGUS.

- fenêtre initiale TCP : la plupart des systèmes d'exploitation utilisent des tailles de fenêtre (presque) constantes dans les paquets réponse. Par exemple, 0x3F25 pour AIX et 0x402E pour MS NT5, OpenBSD et FreeBSD.

- test "Don't fragment bit" : certains systèmes d'exploitation positionnent ce bit dans certains paquets. C'est un autre moyen de déterminer l'OS en service.

-test des options TCP : la base de ce test est simple. Vous envoyez une requête à un hôte spécifique, définissez diverses options et lisez la réponse pour savoir si ces options existent. Les options contenues

dans la réponse sont donc supportées. Selon le système d'exploitation et la version, certaines options ne sont pas admises. C'est un moyen de déterminer le système d'exploitation avec plus de précision.

Paramètre Nmap : -O

Tous les tests ne sont pas présentés, mais il existe de nombreux textes sur Internet concernant le sujet. Fyodor (l'auteur de NMAP) a écrit un article là-dessus, n'hésitez pas à le lire. Ce chapitre n'offre qu'un bref aperçu des techniques de scan les plus répandues. Quelqu'un qui travaille sur la détection d'anomalies de protocoles trouvera plusieurs points de départ (par exemple, certaines combinaisons de drapeaux pouvant être considérées comme anormales).

Autres choses relatives à ICMP
En plus des echo reply/request, ICMP supporte d'autres types de messages grâce auxquels vous pouvez obtenir plus ample information sur le réseau (ce qu'on appelle NON - ECHO - REQUESTS):

ICMP Time Stamp Request / Reply (RFC 792):
Le véritable rôle de Time Stamp Requests (type 13) est d'obtenir les réglages d'heure d'un système distant. S'il reçoit un Time Stamp Request, il renvoie un Time Stamp Reply (type 14). Voici d'abord la structure d'un "time stamp" (selon la RFC):

```
-------------------------------------------------
|         Type  |      Code      |   Checksum     |
-------------------------------------------------
|      Representer     |       Sequence           |
-------------------------------------------------
|              Originate Timestamp                |
-------------------------------------------------
|              Receive Stamp                      |
-------------------------------------------------
|              Transmit Timestamp                 |
-------------------------------------------------
```

Avant d'aborder l'utilisation de Time Stamp Request pour un attaquant voici quelques bases sur les "time stamps". Pour nous, seuls les trois derniers champs sont importants. Voici la partie correspondante de la RFC :

```
L'"Originate Timestamp" est l'heure à laquelle l'expéditeur a modifié pour la
dernière fois le message avant de l'envoyer, le "Receive Timestamp" est l'heure
à laquelle le destinataire l'a modifié dès réception et le
"Transmit Timestamp" est l'heure à laquelle le destinataire l'a modifié avant
envoi."
```

Comme indiqué dans la RFC le "time stamp" renvoyé est le compte en millisecondes depuis minuit UT (GMT).

Quelle est l'utilité d'un time stamp request/reply ?

Si vous recevez une réponse "time stamp", vous savez que l'hôte est accessible et, grâce à l'"Originate Stamp" et au "Receive Stamp" vous pouvez évaluer la charge du réseau (bien sûr, tout dépend du câblage, des cartes...)

Voici une trace de tcpdump :

```
11:38:37.898253 Diablo > Diablo: icmp: time stamp query id 53763 seq 64548
(ttl 254, id 13170, len 40)
                4500 0028 3372 0000 fe01 8b60 7f00 0001
                7f00 0001 0d00 61fb d203 fc24 0211 c0ca
                0000 0000 0000 0000

11:38:37.898253 Diablo > Diablo: icmp: time stamp reply id 53763 seq 64548 :
org 0x211c0ca recv 0x211c0ca xmit 0x211c0ca (DF) (ttl 255, id 0, len 40)
                4500 0028 0000 4000 ff01 7dd2 7f00 0001
                7f00 0001 0e00 db43 d203 fc24 0211 c0ca
                0211 c0ca 0211 c0ca
```

ICMP Information Request / Reply (RFC 792): "Ce message peut être envoyé avec le réseau source dans l'entête IP source et l'adresse de destination avec les champs à zéro (ce qui signifie "ce" réseau). Le module IP doit envoyer la réponse avec les adresses complètes. Ce message est un moyen pour l'hôte de connaître le numéro du réseau sur lequel il se trouve."

Ainsi, une Information Request (type 15) a pour but d'obtenir le numéro du réseau de l'hôte qui a envoyé la requête.

```
L'adresse de la source dans un message de demande d'information sera la
destination du message de réponse. Pour former un message de réponse, les
adresses source et destination sont simplement inversées.
```

Avec une Information Reply (type 16) vous prenez la source IP de l'Information Request comme destination IP de la réponse (autrement dit, vous envoyez la réponse à l'hôte qui a réclamé l'information). Comme source IP de la réponse vous prenez l'IP de destination de la requête.

Normalement, l'expéditeur d'une Information Request définit à 0 l'adresse de destination (qui signifie "ce réseau"). Mais la possibilité existe aussi de définir l'IP destination et l'IP source à 0 (lors de l'envoi de la requête). Dans ce cas, l'Information Reply recevra le numéro du réseau de l'hôte dans le champ adresse de la source et de la destination; par exemple, si le champ de l'adresse source dans la requête n'est pas égal à 0, le numéro du réseau de l'hôte sera seulement renvoyé dans le champ IP source de la réponse.

```
-----------------------------------------------------------------
|         Type  |      Code      |   Checksum          |
-----------------------------------------------------------------
|    Pointer     |       Sequence            |
-----------------------------------------------------------------
```

C'est comme si vous ne pouviez envoyer une demande d'information qu'à l'intérieur d'un même réseau (voir ci-dessus), ce qui n'a pas de raison d'être. Certains systèmes d'exploitation répondent à une demande d'information. Dans une telle réponse on reçoit l'adresse IP de l'hôte et non le numéro du réseau.

Voici un trace tcpdump :

```
11:42:35.608253 Diablo > Diablo: icmp: information request (ttl 255,
```

```
 id 13170, len 28)
                          4500 001c 3372 0000 ff01 8a6c 7f00 0001
                          7f00 0001 0f00 1afc d603 0000
11:42:36.608253 Diablo > Diablo: icmp: information request (ttl 255,
 id 13170, len 28)
                          4500 001c 3372 0000 ff01 8a6c 7f00 0001
                          7f00 0001 0f00 19fc d603 0100
```

ICMP Address Mask Request / Reply (RFC 950): l'Address Mask Request (type 17) a été décrit dans une autre RFC. Pour plus ample information, consultez la RFC950 et non la 792. Le rôle et l'utilisation d'un "Address Mask Request" consiste à obtenir le masque de sous-réseau d'un réseau donné. Si une passerelle reçoit une telle requête elle doit renvoyer l'information correspondante au noeud concerné (Address Mask Reply - type 18).

```
 ------------------------------------------------------------
 | Type  |      Code      |   Checksum                       |
 ------------------------------------------------------------
 | Flag         |       Sequence                            |
 ------------------------------------------------------------
 |               Address Mask                                |
 ------------------------------------------------------------
```

Ceci permet, non seulement de découvrir les hôtes (connectés) du réseau mais aussi la configuration du réseau grâce à quelques tests supplémentaires.

La trace tcpdump :

```
11:45:26.678253 Diablo > Diablo: icmp: address mask request (ttl 254, id
 13170, len 32)
               4500 0020 3372 0000 fe01 8b68 7f00 0001
               7f00 0001 1100 edd7 dc03 2524 0000 0000
```

J'espère que cette partie a montré les nombreuses possibilités destinées à obtenir de l'information sur un réseau sans passer par le ping "habituel". Les différentes RFC décrivent la plupart de ces astuces que vous devez prendre en compte si vous voulez "supporter" les différents types de requêtes. Les développeurs d'IDS devraient aussi considérer ces particularités. Si vous devez les utiliser vous devez vérifier que tout se passe de manière conforme (voir les RFC). Mais ce n'est pas tout comme vous le verrez dans la partie suivante.

Encore plus d'information sur la cible : l'utilisation du filtrage de paquets (plus communément nommé pare-feu) n'est pas vraiment exceptionnelle aujourd'hui. De même l'utilisation de modules pour pare-feu dans les IDS se répand de plus en plus. Souvent, un attaquant ne peut pas utiliser les scans présentés précédemment parce qu'ils sont bloqués, filtrés... Le but de ce bref chapitre est de montrer comment résoudre quelques unes des règles du pare-feu de la cible.

Le principe est simple puisque vous essayez de provoquer des messages d'erreur ICMP en envoyant des paquets "illégaux", ce qui permet déjà d'avoir quelques présomptions sur l'ensemble des règles.

```
Si la passerelle ou l'hôte analysant un datagramme découvre un problème avec les
paramètres de l'entête l'empêchant d'analyser correctement le datagramme, ce
dernier doit être supprimé. Une des sources de ce type de problème peut venir
d'arguments incorrects dans une option. La passerelle ou l'hôte peuvent aussi
informer l'hôte source par l'intermédiaire du message de problème de paramètre.
Ce message n'est envoyé que si l'erreur a abouti à la suppression du datagramme.
```

Cet extrait est tiré de la RFC 792 et fait partie de la description du problème de paramètre (type 12). Une des causes de ce message de problème de paramètre peut être un entête IP erroné, par exemple, si nous envoyons un paquet contenant un entête IP erroné à un hôte, nous devrions recevoir ce message d'erreur. Un autre avantage vient du fait que son support est recommandé dans la RFC 1122 :

```
Un hôte DEVRAIT générer des messages de problème de paramètre. Un message de
problème de paramètre DOIT être passé à la couche transport et il PEUT être
envoyé à l'utilisateur.

DISCUSSION:
Le message de problème de paramètre ICMP est envoyé à l'hôte source si le
problème n'est pas géré par un autre message ICMP. La réception de ce message
indique en général une erreur de configuration distante ou locale.
```

Dans l'ensemble, il s'agit d'un bon moyen de découvrir les hôtes d'un réseau.

On peut en plus se référer à la RFC 1812:

```
 4.3.3.5 Problème de paramètre

Un routeur DOIT générer un message de problème de paramètre pour toute erreur
non gérée par d'autres messages ICMP. Le champ de l'entête IP ou l'option IP
contenant l'octet indiqué dans le champ du pointeur DOIT être inclus dans
l'entête IP renvoyé par ce message ICMP. La section
[4.3.2] décrit l'exception à cette règle.
```

Pourtant, de nombreux routeurs interprètent cette section (et bien d'autres) différemment, ainsi ce problème de paramètre n'est pas toujours notifié.

Un IDS (et un pare-feu) devrait toujours contrôler les champs de l'entête IP puisqu'il peut arriver que vous receviez un paquet à l'entête erroné, même si c'est rare. Un attaquant qui scanne un réseau entier (ou une étendue d'adresses IP) par cette méthode, sait que le pare-feu ne bloque pas ou ne filtre pas ce message d'erreur. Si vous ne recevez pas ce message de problème de paramètre, vous savez au moins qu'il est bloqué.

La découverte d'un jeu de règles par cette méthode n'est qu'une première étape (elle est avant tout une alternative au ping). Pour obtenir une ACL (Liste de Contrôle d'Accès) plus précise de l'éventuel logiciel de filtrage, il faudra plus ample information sur la topologie. Pour ce faire, vous pourriez envoyer différents types de messages ICMP à chaque hôte (avec entête erroné) et attendre un éventuel message de problème de paramètre. Si une notification de problème de paramètre est reçue de l'hôte X, ça signifie que cet hôte ne filtre pas ce type de message ICMP (et bien sûr, que cet hôte est accessible). Ca signifie aussi que l'information erronée dans l'entête IP n'est pas vérifiée. Si nous ne recevons pas ce message, nous pouvons tirer quelques conclusions. D'un côté, il est envisageable qu'un filtre bloque le type de message ICMP, de l'autre, on peut penser que le routeur vérifie l'entête pour empêcher ce genre d'anomalie, etc. Selon les deux résultats vous pouvez déduire certaines règles de filtrage, et enfin, vous obtenez une image révélatrice de ce qui est autorisé et de ce qui ne l'est pas. D'autres possibilités sont offertes par l'utilisation de différents protocoles (TCP, UDP...) afin de découvrir d'autres règles. Par exemple, le fait qu'un protocole ou qu'un port (de l'hôte) soient bloqués ou filtrés peut expliquer l'absence de notification du problème de paramètre.

Je pense que le résumé du problème de paramètre est suffisant pour que nous puissions continuer avec d'autres messages d'erreur.

La partie suivante est extraite de la RFC 1812 et explique ce qu'est l'erreur Destination Unreachable (destination inaccessible) et quelles peuvent en être les causes.

Le message ICMP Destination Unreachable est envoyé par un routeur en réponse à un paquet qu'il ne peut pas acheminer parce que la destination est inaccessible ou que le service n'est pas disponible. Les exemples de tels cas comprennent les messages adressés à un hôte qui n'est pas connecté et qui par conséquent ne répond pas aux requêtes ARP et les messages adressés à des préfixes réseau pour lesquels le routeur ne possède pas de route valide.

Un routeur DOIT être capable de générer des messages ICMP Destination Unreachable et DOIT choisir un code de réponse qui corresponde le mieux possible à la cause de la création de ce message.

0 = Network Unreachable (Réseau inaccessible) - généré par un routeur si aucune route vers le réseau de destination n'est disponible;

1 = Host Unreachable (Hôte inaccessible) - généré par un routeur si aucune route vers l'hôte de destination d'un réseau directement connecté n'est disponible aux requêtes ARP);

2 = Protocol Unreachable (Protocole inaccessible) - généré si le protocole de transport défini dans un datagramme n'est pas supporté par la couche de transport de la destination finale;

3 = Port Unreachable (Port inaccessible) - généré si le protocole de transport défini (UDP par exemple) est incapable de "démultiplexer" le datagramme de la couche de transport de la destination finale et n'a pas de mécanisme pour en informer l'expéditeur;

4 = Fragmentation Needed and DF Set (Fragmentation nécessaire et drapeau DF actif)- généré si un routeur doit fragmenter un datagramme et ne peut y arriver parce que le drapeau DF est actif;

5 = Source Route Failed (Echec de la route source) - généré si le routeur ne peut pas acheminer un paquet vers le saut suivant dans une option de route source;

6 = Destination Network Unknown (Réseau de destination inconnu) - ce code ne DEVRAIT PAS être généré puisqu'il implique côté routeur que le réseau de destination n'existe pas (le code 0 net unreachable (réseau inaccessible) DEVRAIT être préféré au code 6);

7 = Destination Host Unknown (Hôte de destination inconnu) - généré seulement lorsqu'un routeur peut déterminer (selon la couche lien) que l'hôte de destination n'existe pas;

11 = Network Unreachable For Type Of Service (Réseau inaccessible pour le Type de Service) - généré par un routeur si la route vers le réseau de destination ou le Type de Service par défaut ne sont pas disponibles;

12 = Host Unreachable For Type Of Service (Hôte inaccessible pour le Type de Service)- généré si un routeur ne peut pas acheminer un paquet parce que sa (ses) route vers la destination ne correspond pas soit au Type de Service demandé dans le datagramme, soit au Type de Service par défaut (0).

Si nous essayons, par exemple, d’envoyer un paquet à un port qui utilise un protocole n’existant pas, nous devrions recevoir une notification de Destination Unreachable avec un code de valeur 2 (Protocol Unreachable). Dans cet exemple, vous devez aussi savoir quels sont les protocoles admis. Vous les trouverez dans /etc/protocols. Après installation de l’un de mes hôtes, voici à quoi il ressemble :

```
 -------------- /etc/protocols ----------------
# /etc/protocols:
# $Id: protocols,v 1.2 2001/01/29 17:29:30 notting Exp $
#
# Internet (IP) protocols
#
#       from: @(#)protocols     5.1 (Berkeley) 4/17/89
#
# Updated for NetBSD based on RFC 1340, Assigned Numbers (July 1992).
#
# See also http://www.isi.edu/in-notes/iana/assignments/protocol-numbers

ip      0       IP              # internet protocol, pseudo protocol number
#hopopt 0       HOPOPT          # hop-by-hop options for ipv6
icmp    1       ICMP            # internet control message protocol
igmp    2       IGMP            # internet group management protocol
ggp     3       GGP             # gateway-gateway protocol
ipencap 4       IP-ENCAP        # IP encapsulated in IP (officially ‘‘IP’’)
st      5       ST              # ST datagram mode
tcp     6       TCP             # transmission control protocol
cbt     7       CBT             # CBT, Tony Ballardie
egp     8       EGP             # exterior gateway protocol
igp     9       IGP             # any private interior gateway
                                # (Cisco: for IGRP)
bbn-rcc 10      BBN-RCC-MON     # BBN RCC Monitoring
nvp     11      NVP-II          # Network Voice Protocol
pup     12      PUP             # PARC universal packet protocol
argus   13      ARGUS           # ARGUS
emcon   14      EMCON           # EMCON
xnet    15      XNET            # Cross Net Debugger
chaos   16      CHAOS           # Chaos
udp     17      UDP             # user datagram protocol
mux     18      MUX             # Multiplexing protocol
dcn     19      DCN-MEAS        # DCN Measurement Subsystems
hmp     20      HMP             # host monitoring protocol
prm     21      PRM             # packet radio measurement protocol
xns-idp 22      XNS-IDP         # Xerox NS IDP
trunk-1 23      TRUNK-1         # Trunk-1
trunk-2 24      TRUNK-2         # Trunk-2
leaf-1  25      LEAF-1          # Leaf-1
leaf-2  26      LEAF-2          # Leaf-2
rdp     27      RDP             # "reliable datagram" protocol
irtp    28      IRTP            # Internet Reliable Transaction Protocol
iso-tp4 29      ISO-TP4         # ISO Transport Protocol Class 4
netblt  30      NETBLT          # Bulk Data Transfer Protocol
mfe-nsp 31      MFE-NSP         # MFE Network Services Protocol
merit-inp       32    MERIT-INP       # MERIT Internodal Protocol
sep     33      SEP             # Sequential Exchange Protocol
3pc     34      3PC             # Third Party Connect Protocol
idpr    35      IDPR            # Inter-Domain Policy Routing Protocol
xtp     36      XTP             # Xpress Tranfer Protocol
ddp     37      DDP             # Datagram Delivery Protocol
idpr-cmtp       38    IDPR-CMTP       # IDPR Control Message Transport Proto
tp++    39      TP++            # TP++ Transport Protocol
il      40      IL              # IL Transport Protocol
```

```
ipv6    41      IPv6            # IPv6
sdrp    42      SDRP            # Source Demand Routing Protocol
ipv6-route      43    IPv6-Route      # Routing Header for IPv6
ipv6-frag       44    IPv6-Frag       # Fragment Header for IPv6
idrp    45      IDRP            # Inter-Domain Routing Protocol
rsvp    46      RSVP            # Resource ReSerVation Protocol
gre     47      GRE             # Generic Routing Encapsulation
mhrp    48      MHRP            # Mobile Host Routing Protocol
bna     49      BNA             # BNA
ipv6-crypt      50    IPv6-Crypt      # Encryption Header for IPv6
ipv6-auth       51    IPv6-Auth       # Authentication Header for IPv6
i-nlsp  52      I-NLSP          # Integrated Net Layer Security TUBA
swipe   53      SWIPE           # IP with Encryption
narp    54      NARP            # NBMA Address Resolution Protocol
mobile  55      MOBILE          # IP Mobility
tlsp    56      TLSP            # Transport Layer Security Protocol
skip    57      SKIP            # SKIP
ipv6-icmp       58    IPv6-ICMP       # ICMP for IPv6
ipv6-nonxt      59    IPv6-NoNxt      # No Next Header for IPv6
ipv6-opts       60    IPv6-Opts       # Destination Options for IPv6
#       61                      # any host internal protocol
cftp    62      CFTP            # CFTP
#       63                      # any local network
sat-expak       64      SAT-EXPAK       # SATNET and Backroom EXPAK
kryptolan       65      KRYPTOLAN       # Kryptolan
rvd     66      RVD             # MIT Remote Virtual Disk Protocol
ippc    67      IPPC            # Internet Pluribus Packet Core
#       68                      # any distributed file system
sat-mon 69      SAT-MON         # SATNET Monitoring
visa    70      VISA            # VISA Protocol
ipcv    71      IPCV            # Internet Packet Core Utility
cpnx    72      CPNX            # Computer Protocol Network Executive
cphb    73      CPHB            # Computer Protocol Heart Beat
wsn     74      WSN             # Wang Span Network
pvp     75      PVP             # Packet Video Protocol
br-sat-mon      76      BR-SAT-MON      # Backroom SATNET Monitoring
sun-nd  77      SUN-ND          # SUN ND PROTOCOL-Temporary
wb-mon  78      WB-MON          # WIDEBAND Monitoring
wb-expak        79      WB-EXPAK        # WIDEBAND EXPAK
iso-ip  80      ISO-IP          # ISO Internet Protocol
vmtp    81      VMTP            # Versatile Message Transport
secure-vmtp     82      SECURE-VMTP     # SECURE-VMTP
vines   83      VINES           # VINES
ttp     84      TTP             # TTP
nsfnet-igp      85      NSFNET-IGP      # NSFNET-IGP
dgp     86      DGP             # Dissimilar Gateway Protocol
tcf     87      TCF             # TCF
eigrp   88      EIGRP           # Enhanced Interi
```

Pourquoi l'hôte ne renverrait-il pas un Protocol Unreachable (alors que vous utilisez un protocole qui n'existe pas) ? Il peut y avoir deux raisons. Soit,vous communiquez avec une machine sous AIX, HP-UX ou Digital Unix, soit le jeu de règles de l'hôte n'autorise pas l'accès à ces ports. Vérifiez donc d'abord le type d'hôte scanné (par la détection d'empreinte de l'OS, par exemple) ou alors considérez que ces ports sont bloqués ou filtrés.

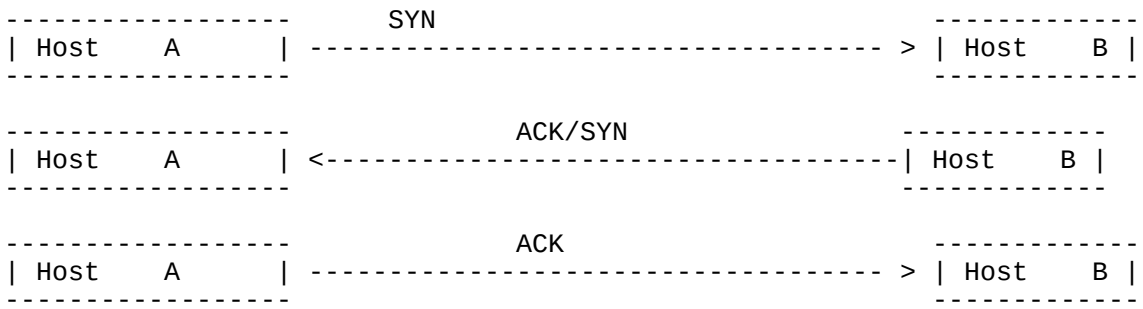
Note: la détection d'une telle attaque est très simple (et appartient au chapitre des détections d'anomalies de protocoles). La détection d'anomalies sera présentée dans le chapitre suivant (possibilités d'analyse). Pour l'instant, il suffit de savoir qu'on recherche des anomalies dans le trafic et que l'utilisation d'un

protocole inexistant fait partie de ces anomalies.

- Denial of Service (Déni de service) (DoS)
Les attaques DoS (Denial of Service) ont pour but de paralyser le serveur cible afin qu'il devienne inaccessible, au moins pour un temps. De très nombreuses techniques existent pour épuiser les ressources d'un hôte cible. Je présente les plus communes dans ce qui suit :

ICMP Flooding :
L'ICMP Flooding "utilise" ICMP. Si un hôte reçoit un echo request ICMP, il essaiera de répondre par un echo reply ICMP. Ce comportement est mis à profit dans l'ICMP Flooding : si vous envoyez à la victime trop d'echo request, elle utilisera ses ressources pour y répondre (par des echo reply). L'IP de l'attaquant étant probablement "empruntée", ce n'est pas lui qui reçoit les réponses mais quelqu'un d'autre.

SYN Flooding:
Afin de bien comprendre le SYN Flooding, je présente de nouveau le three-way-handshake "normal" :



L'hôte A envoie un SYN à l'hôte B pour réclamer une connexion, B répond par un ACK/SYN et attend le ACK final avec lequel la connexion s'établit. Mais que se passe-t-il si le dernier ACK n'est pas envoyé ? Si B renvoie son SYN/ACK, l'ACK de A sera attendu dans la file d'attente de connexion de B. Si la connexion est établie (A a envoyé son ACK à B), la file d'attente sera purgée. Mais comme dans la plupart des cas, l'adresse IP est "empruntée", B ne recevra jamais d'ACK. Il est donc possible de "remplir" la file d'attente de connexion tant que l'hôte ne peut pas communiquer.

UDP Flooding:
Avec cette attaque le serveur cible est "inondé" de paquets UDP. Si vous envoyez un paquet UDP à un port du serveur cible, il cherchera d'abord le service en charge de cette requête. Vous choisissez alors des ports aléatoires vers lesquels vous envoyez les paquets afin d'augmenter la probabilité d'envoi de messages ICMP "Port Unreachable". Un tel afflux de données (les messages) a pour conséquence l'écroulement considérable des performances de ce segment de réseau.

Land:
Plus tard, nous verrons un filtre capable de détecter une attaque "Land" et de prendre des mesures réactives. Mais voyons d'abord l'attaque proprement dite. Une attaque "Land" utilise la même adresse IP pour la source et la destination. Lorsque vous envoyez des paquets SYN (avec la même IP source et destination) vers un port ouvert, il s'établit une "race condition" (condition de concurrence) sur l'hôte victime qui entraîne la paralysie du système. En voici une trace :

```
 23/06/02  23:12:48  194.157.1.1  80  -> 194.157.1.1
```

```
23/06/02  23:14:57  194.157.1.1  31337  ->     194.157.1.1
```

Comme vous pouvez le voir les adresses IP source et destination sont les mêmes. Une autre trace tcpdump :

```
12:35:26.916369 192.168.38.110.135>192.168.38.110.135: udp 46[tos0x3,ECT,CE]
        4503 004a 96ac 0000 4011 15c7 c0a8 266e
       c0a8 266e 0087 0087 0036 8433 6920 616d
       206c 616d 6520 646f 7320 6b69 6420 6275
       7420 6920 7265
12:35:26.916566 192.168.38.110.135>192.168.38.110.135: udp 46[tos0x3,ECT,CE]
       4503 004a 2923 0000 4011 8350 c0a8 266e
       c0a8 266e 0087 0087 0036 8433 6920 616d
       206c 616d 6520 646f 7320 6b69 6420 6275
       7420 6920 7265
12:35:26.916682 192.168.38.110.135>192.168.38.110.135:udp 46[tos0x3,ECT,CE]
       4503 004a 50a0 0000 4011 5bd3 c0a8 266e
       c0a8 266e 0087 0087 0036 8433 6920 616d
       206c 616d 6520 646f 7320 6b69 6420 6275
       7420 6920 7265
```

L’attaque suivante est aussi appelée Snork.

Teardrop:
Ici, on utilise la fragmentation de paquets IP. Comme vous l’avez vu dans la description des scans, une fragmentation existe si les datagrammes sont plus gros que la taille limite, nommée MTU (Maximum Transmission Unit). Dans cette attaque, les fragments se chevauchent, ce qui entraîne (ou a entraîné) le "crash du système avec de nombreux OS.

```
10:13:32.104203 10.10.10.10.53>192.168.1.3.53: udp 28(frag 242:36@0+)(ttl64)
       4500 0038 00f2 2000 4011 8404 0a0a 0a0a
       c0a8 0103 0035 0035 0024 0000 0000 0000
       0000 0000 0000 0000 0000 0000 0000 0000
        0000 0000 0000

10:13:32.104272 10.10.10.10 >192.168.1.3: udp 28(frag 242:4@24)(ttl 64)
       4500 0018 00f2 0003 4011 a421 0a0a 0a0a
       c0a8 0103 0035 0035 0024 0000 0000 0000
       0000 0000 0000 0000 0000 0000 0000 0000
```

Ping of Death:
Ici aussi, la fragmentation de paquets IP joue un rôle important. Je vais approfondir un peu ce qu’est la fragmentation. Comme déjà dit, la fragmentation signifie que la taille des datagrammes est réduite et que des fragments simples ne doivent pas être plus gros que le MTU. Lors d’une fragmentation, vous devez tenir compte du fait que vous ne pouvez pas faire ce que vous voulez : certains champs doivent exister dans chaque fragment. Ainsi, chaque fragment doit contenir l’entête du protocole IP afin de choisir la bonne route. Le routeur peut reconstituer des fragments en un datagramme dans lequel chaque fragment reçoit un drapeau de 16 bits (du "gros" datagramme original). Ce drapeau de 16 bits permet ultérieurement de trier les fragments isolés en un datagramme correct. Il existe, de plus, un décalage qui permet de dire quelle était la position du fragment dans le datagramme original. Mais cette position est indiquée en unités de 8 octets (puisqu’elle est mesurée ainsi). Le bit supplémentaire informe de l’existence d’autres fragments de ce datagramme. S’il est positionné à 1, d’autres fragments existent, s’il est à 0, il s’agit du dernier fragment du datagramme. Voyons maintenant l’attaque "ping of death". On définit un décalage du dernier fragment pour le "ping of death" : décalage + taille du fragment > 65535

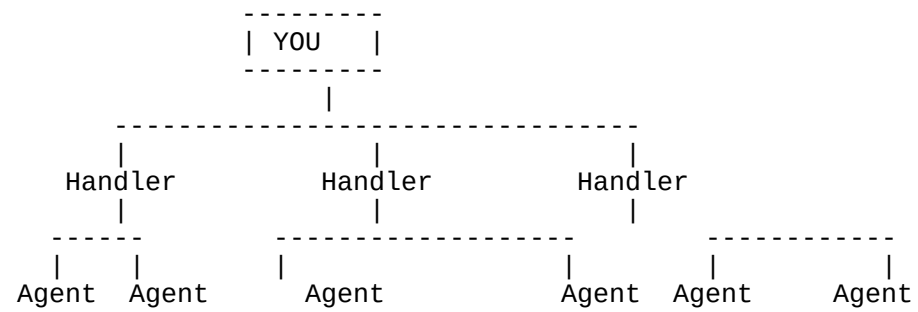
octets. Ainsi, il devient possible d'inclure des variables de 16 bits qui aboutiront, par exemple, au "crash" du système.

```
17:43:58.431 pinger > target: icmp echo request (frag 4321: 380@0+)
17:43:58.431 pinger > target: (frag 4321: 380@2656+)
17:43:58.431 pinger > target: (frag 4321: 380@3040+)
17:43:58.431 pinger > target: (frag 4321: 380@3416+)
```

Smurf:
Avec cette attaque, on envoie un ping à l'adresse de diffusion (broadcast), autrement dit, de nombreux ping sont transmis. Les paquets envoyés à une adresse de diffusion atteignent tous les hôtes du réseau et sont traités sur chacun. Si vous envoyez de nombreux ping (ICMP echo request) à l'adresse de diffusion (par exemple 10000 par seconde) et que le réseau soit d'environ 1000 hôtes, ça signifie qu'il y a 10000 * 1000 réponses ICMP echo reply par seconde sur l'hôte victime, soit 10000000 réponses ICMP echo reply.

```
09:28:28.666073 179.135.168.43>256.256.30.255: icmp: echo request (DF)
       4500 001c c014 4000 1e01 6172 b387 a82b
       c0a8 1eff 0800 f7ff 0000 0000 0000 0000
       0000 0000 0000 0000 0000 0000 0000
09:28:28.696073 68.90.226.250>256.256.30.255: icmp: echo request (DF)
       4500 001c c015 4000 1e01 95cf 445a e2fa
       c0a8 1eff 0800 f7ff 0000 0000 3136 3803
       3133 3503 3137 3907 696e 2d61 6464
09:28:28.726073 138.98.10.247>256.256.30.255:  icmp: echo request (DF)
       4500 001c c016 4000 1e01 27ca 8a62 0af7
       c0a8 1eff 0800 f7ff 0000 0000 0332 3236
       3938 0331 3638 0769 6e2d 6164 6472
09:28:28.756073 130.113.202.100 > 256.256.30.255: icmp: echo request (DF)
       4500 001c c017 4000 1e01 704c 8271ca64
       c0a8 1eff 0800 f7ff 0000 0000 0231 3002
       3938 0331 3338 0769 6e2d 6164 6472
       ...
```

Il existe aussi depuis un certain temps, les attaques DDoS (Distributed Denial of Service ou Déni de service distribué)). Comme le nom l'indique, il s'agit d'une attaque DoS distribuée (par le réseau). L'attaquant (le client), recherche les autres hôtes ou réseaux faciles à exploiter. Les premiers hôtes touchés se nomment les gestionnaires (handlers). Ces derniers "contaminent" d'autres hôtes ou réseaux nommés agents;



Les agents lanceront les attaques ultérieurement.

Translation information:

de --> -- : Klaus Müller <Socma(at)gmx.net>

de --> en: Hubert Kaißer
<hubert(Q)faveve.uni-stuttgart.de>

en --> fr: Georges Tarbouriech <gt(at)linuxfocus.org>

2005-01-14, generated by lfparser_pdf version 2.51