ALTRAN ITALIA per

Sessione di studio AIEA

Policy per la sicurezza di basi dati

ALTRAN ITALIA SPA
Capitale Sociale € 1.000.000 Partita IVA e CF 03932470010
C.C.I.A.A. N.600554 Registro delle Imprese N. 4309/81

Torino – Strada del Drosso, 33/19 Palazzo C 10135 tel. 011-19740700 fax 011-19740701
Roma – Via Terenzio, 35 00193 tel. 06-6879260 fax 06-6879429
Milano – Via De Togni, 29 Milano tel. 02-4335.031 fax 02.4335.0343
Napoli – CD Via Porzio, 4 IS G1 80143 tel 081-2128720 fax 081-2128722

SOMMARIO

1 Introduzione

L'obiettivo di questo documento è quello di suggerire le linee guida di base per la messa in sicurezza di basi dati (DB); è un documento generico e non fa riferimento a specifici RDBMS, ma fornisce alcuni suggerimenti ed esempi relativi ad alcuni di essi.

E' da notare che il modo migliore per implementare una effettiva sicurezza sulle basi dati, è quello di includere nel design iniziale degli applicativi che le utilizzano, i concetti di sicurezza più comuni ed eventualmente, se possibile, scegliere un RDBMS che permetta in modo semplice ed efficace l'implementazione delle policy e delle componenti di sicurezza studiate a priori.

L'ultimo capitolo di questo documento suggerisce anche una metodologia da applicarsi nel caso in cui si debba mettere in sicurezza basi dati già in produzione.

Ovviamente, la messa in sicurezza di una base dati deve essere eseguita su sistemi operativi su cui sia stato preventivamente effettuato il necessario hardening.

2 Documentazione applicazioni e sistemi

Prima di poter operare un hardening efficiente, è necessario che ogni singola applicazione che utilizza una base dati sia catalogata e descritta.

Questa attività dovrebbe essere eseguita a partire dalle prime fasi di definizione di un nuovo progetto, ed essere continuamente mantenuta nel tempo (soprattutto durante il periodo di gestione tecnica in erogazione).

Nello stesso modo devono essere catalogati e descritti i sistemi, intesi solitamente come server, che compongono le architetture applicative.

L'utilizzo di strumenti automatici per la gestione degli asset informatici può essere utile, ma solitamente manca di alcune informazioni che devono comunque essere riportate, per esempio:

- Le procedure di integrazione applicativa (quale base dati è utilizzata da un certo application server)
- La catalogazione della riservatezza dei dati
- Le attività di gestione ordinaria e straordinaria dei sistemi/applicativi
- Le figure che sono responsabili per il corretto funzionamento delle applicazioni o dei sistemi
- Le procedure di gestione degli incidenti

3 Elementi di sicurezza comuni per le basi dati

Ogni RDBMS fornisce diverse funzionalità che hanno impatto sulla sicurezza dei dati che esso gestisce. Questa sezione provvede a descriverle ed a fornire suggerimenti sulla loro configurazione con lo scopo di migliorare il livello di sicurezza dei dati gestiti dalle applicazioni.

3.1 Autenticazione

I database forniscono solitamente un loro sistema di autenticazione, oppure sono in grado di appoggiarsi ad altri sistemi di autenticazione esterna (es. LDAP, AD, token, smart card etc).

Nel caso in cui si utilizzi la base dati stessa per l'autenticazione degli utenti, vi sono in genere tre modalità utilizzate dagli applicativi:

- Utilizzo di un utente RDBMS
- Utilizzo dell'autenticazione da sistema operativo
- Utilizzo di una tabella autenticativa custom

3.1.1 Utilizzo di un utente RDBMS

In termini di sicurezza è una buona soluzione in quanto:

- Le password degli utenti sono già memorizzate in modalità sicura
- E' possibile limitare gli utenti permettendo operazioni soltanto su tabelle/viste ben definite
- Eventuali vulnerabilità legate alla gestione degli utenti sono corrette dal produttore e testate da un elevato numero di utilizzatori

Questa soluzione ha comunque degli svantaggi che devono essere verificati in fase di progettazione dell'applicazione:

- Alcuni RDBMS hanno il costo che è funzione diretta degli utenti registrati sullo stesso
- Aumento della complessità applicativa per la gestione degli utenti e dei ruoli
- Necessità di utenti specifici per l'esecuzione di attività di audit e/o batch

1	Verificare l'utilizzo di utenti di database	Note:
2	Verificare o definire una procedura automatica (es. via batch o meglio con un Identity Manager) per le verifica della congruenza tra gli utenti censiti sul database e quelli che debbono effettivamente accederci (es. non vi siano persone che si sono dimesse o che sono state trasferite ad altra unità lavorativa che non necessita accesso all'applicazione)	Note:
3	Verificare la profilazione degli utenti in modo che i permessi di accesso siano limitati alle sole tabelle/viste necessarie e non sia possibile accedere ad altri DB e/o tabelle di sistema.	Note:

3.1.2 Utilizzo dell'autenticazione da sistema operativo

Con alcuni RDBMS è possibile utilizzare utenti di sistema operativo (dominio AD o NIS); questa è la soluzione migliore in quanto, oltre ai vantaggi che si riscontrano in "utilizzo di un utente RDBMS", è strettamente integrata nell'ambiente operativo e impedisce il transito sulla rete delle coppie login/password necessarie affinché l'autenticazione abbia luogo, utilizzando i meccanismi di autenticazione forniti dal sistema operativo.

Anche questa soluzione ha comunque lo svantaggio, da valutare in fase di progettazione dell'applicazione, di un aumento della complessità applicativa per la gestione degli utenti e dei ruoli definiti per gli utenti nel sistema operativo, in relazione all'utilizzo degli stessi all'interno del RDBMS.

4	Verificare l'utilizzo di utenti di sistema operativo	Note:
5	Verificare o definire una procedura automatica (es. via batch o meglio con un Identity Manager) per la verifica della congruenza tra gli utenti censiti sul sistema operativo (in particolari gruppi) e quelli che debbono effettivamente accederci (es. non vi siano persone che si sono dimesse o che sono state trasferite ad altra unità lavorativa che non necessita accesso all'applicazione)	Note:
6	Verificare la profilazione degli utenti in modo che i permessi di accesso siano limitati alle sole tabelle/viste necessarie e non sia possibile accedere ad altri DB e/o tabelle di sistema.	Note:
7	Verificare le modalità di autenticazione da remoto (fuori dalla rete locale), se previste.	Note:

3.1.3 Utilizzo di una tabella autenticativa custom

Questa soluzione è solitamente pericolosa, in quanto la sicurezza delle credenziali è lasciata agli sviluppatori. I problemi più comunemente riscontrati sono:

- Password degli utenti non cifrate nella tabella di autenticazione
- Non vi possono essere discriminazioni agli accessi sulle tabelle in base agli utenti
- Molto spesso l'applicazione si autentica sulla base dati utilizzando credenziali riservate ai DBA (es. system per Oracle, sa per MS SQL Server, root per MySQL)
- Gli accessi non controllati dall'applicazione (es. via ODBC) avvengono con le credenziali dell'applicazione e quindi con possibilità elevate di causare danni

- Nel caso in cui l'applicazione sia vulnerabile a qualche tipo di attacco (es. SQL Injection) è molto semplice per un hacker ottenere l'elenco di tutti gli utenti e le loro password

L'utilizzo di questo tipo di autenticazione dovrebbe essere limitato a casi specifici:

- Applicazioni di terze parti, leader di mercato o di rilevanza elevata per il business
- Applicazioni datate per cui non è possibile programmare delle modifiche
- Applicazioni per cui sia stata effettuata una completa verifica di sicurezza applicativa e/o una formale code review.

8	Verificare l'utilizzo di password cifrate nella tabella che autentica gli utenti	Note:
9	Verificare che l'applicazione non utilizzi un utente amministratore di base dati (es. system, sa, root) per operare	Note:
10	Verificare o definire una procedura automatica (es. via batch o meglio con un Identity Manager) per la verifica della congruenza tra gli utenti censiti sulla tabella autenticativa e quelli che debbono effettivamente accederci (es. non vi siano persone che si sono dimesse o che sono state trasferite ad altra unità lavorativa che non necessita accesso all'applicazione)	Note:
11	Eseguire una completa analisi di sicurezza sull'applicazione con la verifica delle utenze definite	Note:

3.2 Autorizzazione

L'autorizzazione è la funzionalità che permette la gestione dei privilegi assegnati agli utenti. La maggior parte dei prodotti di RDMBS permette di specificare per ogni utente o profilo, quali sono i suoi privilegi CRUD (create, read, update e delete) in modalità molto fine (tabelle, viste etc).
Inoltre è anche possibile specificare i permessi di esecuzione delle stored procedure.

A volte questa funzionalità è confusa con l'autenticazione, ma mentre quest'ultima è solitamente utilizzata per concedere l'accesso o meno ad un sistema, l'autorizzazione indica quali sono le possibilità che un utente ha di operare sullo stesso (ovvero cosa può in effetti fare).

Come per l'autenticazione, vi sono in genere due possibilità per implementare la fase autenticativa:

- Utilizzo delle funzionalità native del RDBMS
- Utilizzo di tabelle autorizzative custom

Le considerazioni sulla sicurezza delle due soluzioni sono formalmente le stesse della fase di autorizzazione.

12	Verificare che ogni utente sia correttamente profilato con i minimi permessi necessari per l'esecuzione delle funzionalità applicative	Note:
13	Verificare che l'utente possa eseguire soltanto le stored procedure necessarie per il funzionamento dell'applicazione (non quelle di sistema)	Note:
14	Verificare o definire una procedura automatica (es. via batch o meglio con un Identity Manager) per la verifica della congruenza tra gli utenti censiti sulla tabella autorizzativa ed il loro profilo da utilizzatore	Note:
15	Eseguire una completa analisi di sicurezza sull'applicazione con la verifica dei profili utente e code review per verificare l'impossibilità di eseguire escalation dei privilegi	Note:

3.3 Confidenzialità

Molti RDBMS permettono sia la creazione di tabelle il cui contenuto è crittato, sia l'utilizzo di canali di comunicazione client/server cifrati.
Alcuni RDBMS permettono inoltre di crittare anche l'eventuale codice applicativo in esso contenuto (es. stored procedure), garantendo quindi anche un'elevata sicurezza sul lato applicativo.

Queste funzionalità divengono molto importanti quando i dati trattati sono particolarmente critici o importanti per l'azienda.

L'utilizzo della cifratura ha ovviamente anche alcuni inconvenienti:

- Difficoltà di debug applicativo (non è possibile verificare in modo semplice i dati letti o scritti dall'applicazione)
- Difficoltà di recovery nel caso in cui non sia possibile recuperare i dati persi da un backup
- Aumento delle risorse macchina necessarie per la computazione (problemi di performance)

E' quindi necessario utilizzare la funzionalità di cifratura offerta da un database soltanto in alcuni casi specifici, in cui i dati trattati sono catalogati come segreti o se contengono informazioni particolarmente sensibili per l'azienda.

16	Verificare che tutti i dati ritenuti altamente riservati (o segreti), riservati o personali siano memorizzati in modalità cifrata sulla base dati	Note:
17	Verificare che le stored procedure utilizzate per il trattamento di dati altamente riservati, riservati o personali siano memorizzate in modalità cifrata sulla base dati	Note:

3.4 Integrità

Questa è una funzionalità fornita da tutti i RDBMS, ma a volte utilizzata solo parzialmente negli sviluppi applicativi.

Il concetto di transazione è ben noto agli sviluppatori, meno noto, o per meglio dire, meno utilizzato è il concetto di "constraint". Molto spesso, relazioni di "on delete cascade" o di integrità referenziale in generale non sono utilizzate.

Durante la fase di progettazione di un applicativo custom è molto importante progettare la base dati per utilizzare queste funzionalità, sia per garantire l'integrità dei dati presenti sul database rispetto ad una singola applicazione, sia nel caso in cui sulla stessa base dati insistano più applicazioni diverse.

L'utilizzo delle funzionalità di integrità referenziale comporta anche alcuni svantaggi, primo di tutti un aumento della complessità in fase di progettazione e sviluppo, ma i vantaggi relativi alla robustezza della base dati sono sicuramente elevati.

18	Verificare l'utilizzo dei sistemi di integrità referenziale della base dati	Note:
19	Verificare che l'applicazione gestisca correttamente le transazioni sulla base dati e non vi siano dati inconsistenti	Note:

3.5 Audit

I diversi RDBMS permettono di attivare funzionalità di audit per l'esecuzione di operazioni specifiche o privilegiate, registrando all'interno di un apposito file di log tali attività.

E' necessario definire a livello di progettazione ed implementare, almeno negli ambienti di produzione e test, tali funzionalità e soprattutto fare in modo di effettuare un backup dei file di log creati dal database.

L'audit delle operazioni è fondamentale non solo per la sicurezza, ma anche per le attività di debug applicativo.

Le funzionalità di audit possono anche essere utilizzate nel caso in cui si debba procedere all'hardening della base dati di un sistema già in produzione. In tal caso, l'attivazione di tali funzionalità deve essere eseguita sull'intera istanza in analisi e ridotta alle sole operazioni privilegiate o specifiche per l'istanza al termine dell'analisi.

20	Verificare la sincronizzazione del sistema operativo con un time server per eventuali correlazioni dei log di audit tra la base dati, l'application server ed altri componenti dell'architettura (es. firewall)	Note:
21	Verificare l'attivazione, il funzionamento ed il corretto backup del log di audit della base dati	Note:
22	Verificare l'attivazione, il funzionamento ed il corretto backup del log di audit dell'applicazione	Note:

4 Elementi di sicurezza applicativa

In questa sezione saranno analizzati alcuni elementi relativi alla sicurezza delle basi dati, dal punto di vista applicativo, fornendo suggerimenti per ottenere una migliore sicurezza.

4.1 Gestione delle credenziali di accesso

Tutti gli applicativi che necessitano di accedere ad una base dati hanno bisogno delle credenziali di accesso.

Tali credenziali, possono essere sia definite sulla base dati stessa utilizzando il concetto di utente proprio del prodotto RDBMS sia essere memorizzate in un file (es. file di configurazione) oppure utilizzare l'autenticazione fornita dal sistema operativo stesso (uso del profilo utente con cui gira l'applicazione).

Nel primo caso l'applicazione richiederà le credenziali all'utente e le userà per accedere alla base dati. E' una situazione ottimale in quanto è necessario solamente verificare che la coppia user e password transiti cifrata sulla rete.

Nel secondo caso invece, l'applicazione dovrà utilizzare una coppia user e password memorizzata da qualche parte. In questa situazione è molto importante verificare che:

- Le credenziali di accesso siano memorizzate cifrate con un algoritmo sicuro
- Il file o il repository che contiene le credenziali non sia accedibile da persone non autorizzate (protezione a livello del sistema operativo)

- Le credenziali non siano memorizzate in chiaro in alcun tipo di file (es. dll, file compilati flash, applet java compilati ecc.)

Nel terzo caso sarà direttamente il database a richiedere le credenziali di autenticazione utente al sistema operativo. E' la soluzione ottimale in quanto la coppia user/password non transita sulla rete.

23	Verificare che le credenziali non siano passate in chiaro durante la connessione con la base dati	Note:
24	Nel caso in cui risulti necessario memorizzare le credenziali di accesso alla base dati, verificare che queste siano cifrate nel file o repository che le contiene e che non sia accedibile da personale non autorizzato	Note:
25	Verificare che nel codice applicativo (compilato o meno) non compaiano le credenziali di accesso in chiaro	Note:

4.2 Protezione da attacchi di SQL Injection

La SQL Injection è una delle tecniche più comuni di attacco alle applicazioni che utilizzano basi dati. Questo tipo di attacco consiste nel tentare l'esecuzione di SQL non autorizzato inserendo nei campi dell'applicazione stringhe che alterano il codice in esecuzione.

Supponendo per esempio di avere una applicazione con due campi, $utente e $password, utilizzati per eseguire l'autenticazione su una tabella applicativa.
Nel caso in cui il codice applicativo, per eseguire l'autenticazione, utilizzi una query SQL così strutturata:

SELECT id FROM tabella_utenti WHERE user='$utente' AND password='$password'

Un hacker potrebbe tentare di inserire nei due campi applicativi i seguenti valori:

$utente -> **admin' --**
$password -> **xxx**

La query che ne risulta diverrebbe:

SELECT id FROM tabella_utenti WHERE user='**admin' --**' AND password='**xxx**'

In pratica si otterrebbe una query che ricerca solamente un utente e lo autentica indipendentemente dalla password.

E' necessario che ogni applicazione sia correttamente scritta per evitare tale problema (e non solo nei campi utilizzati per l'autenticazione).

In pratica si deve:

- Controllare tutti i caratteri ricevuti in input ed eliminare quelli non concessi
- Sostituire i caratteri pericolosi con gli equivalenti quotati (es. il singolo apice deve essere ripetuto)
- Forzare o non accettare dati che non rispettino il formato atteso in input

Tali controlli devono necessariamente essere eseguiti a livello server, in quanto l'implementazione di questi a livello del client può sempre essere semplicemente bypassata da un hacker. I controlli devono essere eseguiti sia per eventuali dati inseriti da un utente, sia per qualsiasi tipo di dato che l'applicazione accetta dall'esterno (es. file da importare, parametri ricevuti da un web service etc.)

26	Controllare che l'applicazione analizzi tutti i caratteri ricevuti in input prima di costruire con gli stessi qualsiasi tipo di query sulla base dati	Note:

4.3 DBLink

Alcuni RDBMS permettono la creazione di "link" tra diverse istanze di base dati, istanze che possono essere sia locali (sulla stessa macchina) sia remote (su macchine diverse).

L'utilizzo di questa funzionalità può essere pericoloso dal punto di vista della sicurezza in quanto:

- L'integrità referenziale dei dati può essere difficoltosa se eseguita tra due basi dati diverse (e quindi rischia di non essere implementata)
- La compromissione di una base dati da parte di un hacker può comportare rischi per la base dati collegata

E' quindi necessario porre particolare attenzione nell'uso di questa caratteristica e verificarne la reale necessità al momento della progettazione applicativa.

27	Verificare le modalità di mantenimento dell'integrità dei dati (solitamente implementate applicativamente)	Note:
28	Verificare le modalità di memorizzazione delle credenziali utilizzate per la creazione del link (queste non devono essere rese leggibili o modificabili da utenti non amministratori della base dati)	Note:
29	Verificare che le credenziali di autenticazione fra le basi dati siano ristrette al solo DBLink	Note:

4.4 ODBC ed accessi diretti

L'utilizzo di ODBC o altri sistemi di accesso diretto alla base dati è solitamente sconsigliato, in quanto impedirebbe un controllo sugli accessi basato su sistemi di filtering (come i firewall per esempio).

Ovviamente l'accesso via ODBC o diretto deve essere concesso per applicazioni client/server a due livelli, a meno di utilizzare sistemi di accesso remoto applicativo (es. Citrix, Tarantella o Microsoft RDP).

L'accesso diretto dovrebbe essere consentito e limitato solo esclusivamente agli amministratori delle basi dati (DBA) e da postazioni/utenze ben identificate.

30	Verificare e limitare l'accesso diretto alle basi dati mediante ODBC o altri strumenti	Note:
31	Identificare e documentare gli eventuali accessi diretti necessari per il personale autorizzato (DBA)	Note:

5 Elementi di sicurezza aggiuntivi per le basi dati

In questa sezione saranno analizzati altri elementi relativi alla sicurezza delle basi dati, fornendo suggerimenti ulteriori per la loro messa in sicurezza.

5.1 Installazione

La maggior parte degli RDBMS installa automaticamente (o su richiesta durante la procedura di installazione stessa) alcune componenti/configurazioni che devono essere riviste manualmente.

Solitamente si tratta di:

- Utenti amministrativi di default (es system/manager su Oracle)
- Database di esempio (es. test su MySQL)
- Utenti di esempio (es. scott/tiger su Oracle)
- Componenti aggiuntive per la gestione o gli sviluppi applicativi

Tali componenti/configurazioni devono essere riconfigurate, nel caso si tratti di utenti con password di default, oppure eliminate, come i database di esempio e gli utenti di esempio.

E' necessario che, per ogni tipologia di database, il DBA incaricato effettui tutte le verifiche necessarie alla corretta configurazione della base dati che sarà utilizzata e che tutte le caratteristiche o funzionalità non necessarie siano rimosse, arrestate (es. listener di Oracle se non necessario) oppure correttamente configurate.

32	Verificare la rimozione delle eventuali istanze di esempio	Note:
33	Verificare l'eliminazione degli eventuali utenti di esempio	Note:
34	Verificare il cambio password per le utenze di default (es. system su Oracle)	Note:
35	Verificare la corretta configurazione delle componenti di sicurezza (es. audit log)	Note:

5.2 Networking

La maggior parte degli RDBMS è dotata di un listener TCP/IP che permette la creazione di architetture client/server a due o tre livelli. In tali architetture la base dati non è installata sulla macchina in cui è invece presente la componente applicativa che la deve utilizzare, e quest'ultima necessita quindi di aprire una comunicazione verso il listener TCP/IP per poter operare correttamente.

In alcuni casi, soprattutto nelle architetture che utilizzano come front-end una applicazione web, tale listener dovrebbe essere solo accessibile dall'application server.

E' comunque sconsigliato che tale listener sia accedibile da qualsiasi altro server/workstation, e per questo motivo è suggeribile:

- Configurare (se possibile) il listener del prodotto RDBMS per essere disponibile solo per determinate macchine (gli application server o altri database nel caso di basi dati distribuite)
- In alternativa utilizzare un firewall hardware o software per limitare l'accesso al listener stesso

Alcuni RDBMS permettono anche di specificare, a livello utente, se è permesso soltanto un login locale (da localhost) oppure anche da remoto. E' necessario configurare correttamente questa caratteristica.

36	Nel caso in cui l'application server sia installato sulla stessa macchina della base dati è necessario arrestare il listener del DB oppure filtrarlo mediante firewall locale o regole sul TCP	Note:
37	Nel caso in cui l'application server non sia installato sulla stessa macchina della base dati, limitare l'accesso al listener, in modo che solo l'application server (ed eventuali altri server	Note:

	applicativi) possa accedervi mediante firewall locale o regole sul TCP	
38	Nel caso in cui il software RDBMS permetta di specificare se l'utente è locale oppure remoto verificarne la configurazione (anche se si utilizza un firewall locale o regole di filtro sul TCP)	Note:

5.3 Processi batch di integrazione/gestione

Molto spesso, in ambienti complessi, vi è la necessità di realizzare degli script che provvedono ad esportare o importare dati da/su RDBMS. Tali script possono essere utilizzati sia in modalità attended, ovvero lanciati da un operatore, sia in modo unattended, ovvero lanciati da uno schedulatore o ad evento da strumenti automatici.

L'utilizzo di script attended o unattended a cui non si sia prestata la necessaria attenzione a livello di sicurezza è un rischio, in quanto:

- Contengono credenziali valide per accedere alla base dati (e molto spesso sono credenziali di alto livello)
- Non sono protetti, a livello di sistema operativo, permettendo a chiunque (o comunque a personale non autorizzato) di modificarli
- Le directory in cui sono memorizzati i file estratti o da importare non sono protette a livello di sistema operativo (permettendo quindi la lettura/scrittura dei dati stessi)

E' quindi necessario per tali script:

- Utilizzare credenziali con i minimi permessi possibili
- Proteggere correttamente tutte le risorse utilizzate (script e file esportati o da esportare)
- Limitare le utenze o gli strumenti automatici che possono eseguire gli script

39	Utilizzare credenziali con i minimi permessi necessari per l'esecuzione degli script	Note:
40	Proteggere lo script in modo che soltanto le persone autorizzate possano accedervi in lettura/scrittura	Note:
41	Proteggere lo script in modo che soltanto persone autorizzate o strumenti automatici possano eseguirlo	Note:
42	Utilizzare directory diverse per contenere gli script e per ospitare i dati esportati o da importare	

43	Proteggere le directory in cui si trovano i dati esportati o da importare in modo che soltanto persone o strumenti automatici possano accedervi	Note:

5.4 Monitoring

I sistemi di monitoring sono una componente importante per la gestione e l'efficienza dei sistemi di produzione. Molto spesso però possono essere anche fonte di elevati problemi di sicurezza, in quanto per poter eseguire alcuni test di verifica (es. accessibilità di una base dati), necessitano di credenziali valide sul servizio da controllare.

E' quindi necessario mettere in sicurezza il sistema di monitoring, ovvero:

- Utilizzare credenziali con i minimi permessi (es. soltanto una select count su una tabella associativa)
- Filtrare gli accessi al listener della base dati permettendo connessioni soltanto dal server di monitoring
- Proteggere il server di monitoring in modo che la configurazione dello stesso non sia accedibile
- Limitare l'eventuale accesso all'agente SNMP della macchina database al solo server di monitoring mediante opportuna configurazione. Non utilizzare i nomi di community standard (public e private) oppure attivare l'autenticazione (SNMP v3)

44	Utilizzare credenziali con i minimi permessi necessari per l'esecuzione dei test	Note:
45	Limitare l'accesso al listener del database, in modo che solo il server di monitoring possa accedervi mediante firewall locale o regole sul TCP	Note:
46	Limitare l'accesso agli agenti SNMP eventualmente presenti sulle macchine che ospitano le basi dati in modo che solo il server di monitoring possa accedervi	Note:
47	Verificare l'utilizzo di community non standard oppure l'utilizzo dell'autenticazione avanzata sugli agenti SNMP	Note:

5.5 Backup

Tutte le basi dati sono solitamente poste sotto backup per implementare un efficiente sistema di disaster recovery.

L'utilizzo di un sistema di backup è solitamente necessario, ma è anche necessario verificarne la sua configurazione per garantire la sicurezza dei dati, soprattutto è necessario porre particolare attenzione nella gestione dei media che contengono le copie di backup, per evitare che queste possano essere accedute da personale non autorizzato.

Nella configurazione dei sistemi di backup è quindi necessario:

- Verificare che i dati salvati dal sistema di backup risiedano solamente sui media predisposti per tale attività (ovvero non rimangano copie dei dati sul server di backup o sulla macchina che ospita la base dati stessa)
- Nel caso in cui si utilizzino sistemi di backup a caldo implementati su una istanza sincronizzata dalla base dati originale, questa non deve essere accedibile da personale non autorizzato

48	Verificare il trattamento e la gestione dei media su cui sono salvate le copie di backup dei dati	Note:
49	Verificare che il sistema di backup non utilizzi copie intermedie (o che queste siano correttamente eliminate al termine del backup)	Note:
50	Nel caso si utilizzino copie di basi dati sincronizzate per l'esecuzione dei backup, queste non devono essere accedibili da personale non autorizzato	Note:

6 Procedure per la messa in sicurezza di basi dati

In questo capitolo sono presentati alcuni suggerimenti relativi alla definizione ed alla implementazione delle policy di sicurezza per le basi dati.

6.1 Definizione delle policy

Molti specialisti di sicurezza IT affermano che la sicurezza dipende principalmente dalle policy dichiarate più che dall'uso di soluzioni tecniche, in realtà è ovvio che definire delle ottime procedure di sicurezza su carta e non divulgarle agli utenti e/o responsabili tecnici è perfettamente inutile.

Sono inoltre necessari strumenti tecnici che aiutino la gestione e la verifica della corretta implementazione delle policy da applicare, per rendere più efficace la gestione della sicurezza.

Le indicazioni, relative alla sicurezza delle basi dati contenute nei capitoli precedenti, sono un buon punto di partenza per la stesura di policy tecniche appropriate, ma devono essere estese per coprire particolarità e funzionalità specifiche per ogni RDBMS utilizzato (es. Oracle, MS SQL, MySQL etc).

Nella definizione delle policy è necessario:

- Documentare correttamente tutte le componenti infrastrutturali ed applicative
- Valutare i rischi associati alla riservatezza dei dati, ovvero catalogare ogni policy in base alla necessità di implementazione rispetto al valore dei dati da proteggere
- Identificare modalità e tempi per l'esecuzione di audit sulla corretta implementazione delle policy (es. definizione di audit interni periodici)
- Individuare strumenti tecnici di controllo sull'implementazione delle policy per evidenziare in tempi ridotti eventuali problemi di sicurezza (es. utilizzare strumenti HIDS per la verifica delle configurazioni)

6.2 Diffusione delle policy

Le policy di sicurezza devono ovviamente essere distribuite e portate a conoscenza di tutti gli interessati, ma è anche necessario fare in modo che queste:

- Vengano applicate
- Siano ricordate dagli interessati

Per una efficiente diffusione è suggeribile quindi:

- Organizzare seminari di presentazione delle policy per tutti gli interessati
- Pubblicare in modo facilmente accessibile le policy stesse (es. portale interno)
- Definire, soprattutto nel momento della prima diffusione, una forma di supporto diretto gestito da personale competente (es. casella di posta, help desk etc)
- Provvedere con cadenza regolare (3 mesi/6 mesi) ad effettuare un incontro o riunione di aggiornamento delle policy, per verificare eventuali nuove implementazioni o correzioni
- Utilizzare gli audit interni o esterni come momento di verifica ed ulteriore diffusione (es. presentazione dei risultati dell'audit a tutti gli interessati)

Le modalità di diffusione devono far parte delle policy stesse, ovvero essere descritte al loro interno, in modo da tenere alta l'attenzione per tutti gli interessati.

6.3 Implementazione delle policy

La fase più critica ed importate nell'introduzione delle policy di sicurezza è quella dell'implementazione, soprattutto nel caso in cui si cerchi di ovviare a problemi di sicurezza già esistenti, in quanto, molto spesso, non esistono ambienti di test su cui provare le modifiche tecniche richieste e queste sono direttamente implementate su ambienti di produzione.

Una corretta procedura per l'implementazione delle policy per la gestione sicura delle basi dati deve provvedere a:

1. documentare tutte le infrastrutture (solitamente server) che compongono le architetture da porre sotto policy
2. documentare tutte le applicazioni e tutti i flussi di integrazione e gestione che interagiscono con le basi dati
3. individuare le policy da adottare basandosi su:
    - livello di riservatezza dei dati

- impatti sulle applicazioni e sugli utilizzatori
- impatti sui sistemi di integrazione/gestione

4. verificare la documentazione e l'analisi di impatto effettuata "a tavolino" utilizzando strumenti tecnici (es. audit degli accessi alla base dati, sniffer di rete etc.)

Solitamente l'implementazione delle policy risulta meno rischiosa se si applica la metodologia del "divide et impera", ovvero se, dall'intero parco delle basi dati, si esegue una prima catalogazione per importanza (dei dati e per il business) e complessità tecnologica, definendo quindi piccoli sotto-insiemi di basi dati su cui le policy possono essere implementate in modalità graduale.

Con questa semplice metodologia è quindi possibile operare per piccoli passi, applicando le modifiche alle basi dati, verificando nel contempo il funzionamento sia delle applicazioni che le utilizzano, sia dei sistemi di integrazione e gestione.