Segurança por Omissão de Sistemas de Gestão de Bases de Dados [1]

Carlos Lourenço[1,2], Miguel Correia[1]

[1] LASIGE, Faculdade de Ciências da Universidade de Lisboa
Campo Grande, Edifício C6, 1749-016 Lisboa, Portugal
[2] Câmara Municipal de Lisboa
Campo Grande 27, 1749-099 Lisboa, Portugal
carlos.lourenco@cm-lisboa.pt mpc@di.fc.ul.pt

Resumo

Os sistemas de gestão de bases de dados relacionais constituem a pedra basilar dos sistemas de informação actuais. O elevado valor económico desses sistemas, conjugado com o elevado risco a que estão expostos quando ligados à Internet, tornam a questão da sua segurança premente. No entanto, frequentemente os administradores de sistemas descuram a segurança desses sistemas. Por essa razão, é fundamental que esses sistemas forneçam *segurança por omissão*, ou seja, que sejam seguros *out of the box*, sem configuração adicional. Este artigo apresenta um estudo comparativo da segurança por omissão de seis sistemas de gestão de bases de dados comerciais e de código aberto. O estudo foi baseado em análise documental e, sobretudo, no teste desses sistemas usando um conjunto de ferramentas de *software*.

1 Introdução

Os *sistemas de gestão de bases de dados* (SGBDs) relacionais constituem a pedra basilar dos sistemas de informação actuais. O elevado valor económico desses sistemas, conjugado com o elevado risco a que estão expostos quando ligados à Internet [18][3], tornam a questão da sua segurança premente. No entanto, frequentemente os administradores de sistemas descuram a configuração dos mecanismos de segurança desses sistemas [12]. As pessoas que projectam *software* muitas vezes consideram que basta fornecer os mecanismos de segurança que permitam proteger o *software* e documentar a sua utilização, para que depois os administradores de sistemas os configurem correctamente. No entanto, Howard e LeBlanc fruto da sua experiência de muitos anos referem 5 razões pelas quais essa abordagem não funciona [6]: os administradores não sabem o que configurar; não sabem como configurar; não sabem o que acontecerá se não configurarem; o sistema está estável – para quê mexer?; não têm tempo.

Esse conjunto de razões justifica a necessidade de os sistemas informáticos em geral, e de os sistemas de gestão de bases de dados em particular, forneçam *segurança por omissão*, ou seja, que sejam seguros *out of the box*, sem configuração adicional. Quando um sistema é instalado, mesmo que o administrador de sistemas não tome especial cuidado com a configuração do sistema, este tem de ficar "suficientemente seguro". Uma questão típica é a das palavras-chave por omissão, uma praga que afecta não apenas SGBDs mas inúmeros outros sistemas, incluindo vários modelos de *routers.* Uma simples procura na Internet do termo "default passwords" permite encontrar as palavras-chave por omissão de inúmeros sistemas e serviços. Os administradores de sistemas têm a possibilidade de modificar essas palavras-chave, removendo assim essas vulnerabilidades, mas o facto é que muitas vezes não o fazem.

O objectivo deste artigo consiste em apresentar um estudo comparativo da segurança por omissão de um conjunto alargado de SGBDs, tanto comerciais como de código aberto: *IBM DB2* (versões 8.2 e 9.1), *Microsoft SQL Server* (2000 e 2005), *MySQL* (4.1 e 5.0), *Oracle* (9i e 10g), *PostgreSQL* (7.4.14 e 8.2), *Sybase* (12.5 e 15). Este estudo permite tirar ilações sobre o que deveria ser a segurança por omissão desses sistemas. Os SGBDs foram escolhidos por serem os mais utilizados actualmente, segundo um estudo da *Gartner* [4].

[1] Este trabalho foi parcialmente financiado pela FCT através da unidade de investigação LASIGE.

Metodologia O estudo pretendeu avaliar a segurança por omissão em termos dos atributos clássicos de segurança: confidencialidade, integridade e disponibilidade, do serviço e dos dados. Mais concretamente, foi avaliado um conjunto de aspectos que podem ser classificados em termos de 3 tipos de mecanismos de segurança:

- *Autenticação:* assegurar que – por omissão – a comunicação do par *{utilizador, palavra-chave}* não é realizada em claro na rede e que existem regras que definem o comprimento mínimo e a complexidade da palavra-chave, sendo obrigatória a definição de palavras-chave para utilizadores criados por omissão pelo SGBD aquando da sua instalação;
- *Autorização:* assegurar que existe um controlo granular no acesso a objectos e que as permissões têm que ser atribuídas aos utilizadores explicitamente;
- *Auditoria de eventos:* assegurar que os acessos a objectos são passíveis de serem monitorizados e descriminados numa estrutura que poderá ser utilizada posteriormente como forma de análise de segurança.

O estudo foi baseado na análise documental e, sobretudo, no teste desses sistemas usando um conjunto de ferramentas de *software*. As ferramentas para auditoria de segurança são múltiplas, sendo algumas generalistas e outras direccionadas a produtos específicos. As ferramentas generalistas utilizadas foram as seguintes:

- Varrimento de portos: *nmap* [27]
- Análise de assinaturas de aplicações: *amap* [30]
- Análise de protocolos: *wireshark* [32]
- Injecção de ataques de negação de serviço: *hping* [9]

As ferramentas para produtos específicos utilizadas foram:

- Varrimento de servidores *web*: *Nikto* [26]
- Ataque de dicionário à autenticação do Oracle: *checkpwd* [20]
- Bibliotecas *open source* para *SQL Server*: *FreeTDS* [23]
- Cliente *IBM DB2*: *EMS SQL Query for DB2* [22]
- Clientes *Microsoft SQL Server*: *osql Query Analyzer* [24]
- Cliente *MySQL*: *EMS MySQL Client* [21]
- Cliente *Oracle*: *SQL*Plus* [28]
- Cliente *PostgreSQL*: pgAdmin III [29]
- Injector de comandos *Oracle* TNS: Tnscmd [31]

Contribuições As contribuições do artigo são as seguintes:

- a avaliação da segurança por omissão de um conjunto alargado de SGBDs comerciais e de código aberto, apresentando com detalhe a avaliação realizada ao *Microsoft SQL Server 2000* e *2005*;
- um conjunto de conclusões sobre como deve ser a segurança por omissão de um SGBD;
- uma análise de como devem ser configurados os SGBDs para ficarem mais protegidos do que por omissão, e das vulnerabilidades que permanecem por não existirem mecanismos para as remover.

Organização O artigo está organizado da seguinte forma. A secção 2 discute o trabalho relacionado. A secção 3 ilustra a forma como foi feito o estudo, reportando a análise do *Microsoft SQL Server* (versões 2000 e 2005), atendendo a que não é possível aqui detalhar a análise de todos os SGBDs. A secção 4 apresenta a comparação dos diversos SGBSs estudados, discute como os configurar, as vulnerabilidades que persistem, e tira conclusões sobre deve ser a segurança por omissão desses sistemas. A secção 5 conclui o artigo.

2 Trabalho relacionado

Num artigo clássico de 1975, Saltzer e Schroeder recolheram um conjunto de *princípios de projecto* para protecção de sistemas informáticos que continuam a ser plenamente válidos [16]. A noção de segurança por omissão está implícita em vários desses princípios. O princípio dos *fail-safe defaults* diz que a omissão é a não permissão de acesso, e que os mecanismos de protecção indicam as condições sob as quais o acesso é permitido. Trata-se pois da segurança por omissão no contexto do controle de acesso. O princípio do *privilégio mínimo* diz que cada programa e cada utilizador devem operar usando o conjunto mínimo de privilégios necessários para realizarem o seu trabalho. O princípio está relacionado com a segurança por omissão em termos dos privi-

légios fornecidos. Um terceiro princípio, o da *aceitabilidade psicológica,* afirma que os mecanismos de segurança têm de ser simples de usar e devem fazer aquilo que o utilizador espera deles. Se considerarmos que um administrador de sistemas tem tendência a considerar que por omissão o sistema está seguro, a segurança por omissão é também uma consequência deste princípio. Em resumo, podemos dizer que a segurança por omissão é um corolário destes três princípios.

Apesar destes princípios serem bem conhecidos, tanto a investigação em segurança como a engenharia de *software* geralmente preocupam-se com fornecer os mecanismos necessários para proteger os sistemas, e menos com a forma de os configurar correctamente, ou de os ter configurados correctamente por omissão. No entanto, existem algumas excepções. Desde que surgiu, o sistema operativo *OpenBSD* primou pela segurança [19]. Um dos elementos que contribui para o seu troféu de "sistema operativo seguro" foi o facto de após a instalação ter todos os serviços desligados, exceptuando o serviço de acesso remoto SSH (*Secure Shell*). Estando os serviços desligados por omissão, o sistema está inicialmente protegido contra vulnerabilidades que esses serviços possam ter. Nos dias de hoje, muitos produtos continuam sem providenciar uma configuração *out-of-the-box* que possamos caracterizar de segura. No entanto, fabricantes como a Microsoft, HP, Oracle ou Sun têm manifestado preocupação em relação a essa questão.

A Microsoft levou esta preocupação a um novo limiar através da iniciativa *Trustworthy Computing*, a partir da qual introduziu no seu ciclo de desenvolvimento de *software* uma estratégia denominada por SD3, constituída pelos seguintes princípios [6]: (a) *Secure by Design* – o *software* deverá ser projectado e concretizado de modo a garantir as propriedades de segurança aplicáveis, por exemplo, resistindo a ataques e garantindo a confidencialidade da informação por ele armazenada; (b) *Secure by Default* – imediatamente após a sua instalação, o *software* deve ficar "suficientemente seguro"; (c) *Secure in Deployment* – o sistema deve ser fácil de manter seguro, ou seja, devem existir ferramentas que permitam aos administradores manter o sistema seguro durante o seu tempo de vida (por exemplo, aplicando *patches*). O segundo princípio é o estudado neste artigo. A estratégia SD3 mais tarde foi denominada de SD3+C, para tornar explícita a necessidade de proteger também a comunicação (daí o "C") [8].

3 Microsoft SQL Server

Esta secção descreve o estudo realizado sobre o SGBD *Microsoft SQL Server*, começando com uma descrição da sua arquitectura e principais componentes, e seguindo-se uma análise da segurança da configuração por omissão das versões 2000 e 2005. O objectivo da secção consiste em ilustrar a forma como foi feito o estudo para *todos* os SGBDs, mas limita-se ao *SQL Server* já que não é possível aqui detalhar a avaliação dos restantes por falta de espaço.

3.1 Modelo de ameaças

Antes de começarmos a avaliar a segurança de um produto é importante desenvolvermos um modelo de ameaças que possa guiar de uma forma organizada o processo de auditoria de segurança. De forma a criarmos o modelo de ameaças é preciso descriminar as ameaças que queremos cobrir, sendo um bom ponto de partida a enumeração dos pontos de entrada do produto [2]. Idealmente deveríamos testar tudo, sem excepção. No entanto, não conseguimos testar minuciosamente todas as componentes de um *software* complexo, necessitamos de estabelecer prioridades. A representação gráfica através de *diagramas de fluxo de dados* (DFDs) permite-nos a identificação das áreas sobre as quais deveremos focar os nossos esforços e o nosso tempo, bem como proporcionar uma melhor compreensão do funcionamento das funcionalidades que pretendemos testar. Por exemplo, o DFD da Figura 1 representa o processo de submissão de credenciais para acesso a um SGBD.

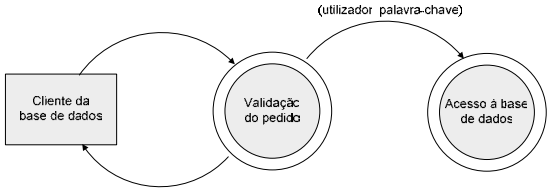

Figura 1 – DFD do processo de submissão de credenciais de acesso a um SGBD

3.2 Arquitectura

Esta subsecção descreve os principais componentes do sistema de gestão de bases de dados *Microsoft SQL Server*, explica a implementação dos mecanismos de Autenticação, Autorização e Auditoria pela Microsoft e faz uma pré-avaliação documental da segurança por omissão.

Uma *instância* é um ambiente lógico, criado dentro de um sistema *SQL Server*, providenciando um intervalo de memória, políticas de gestão de CPU e portos para comunicação. Cada instância é composta por várias bases de dados. Cada base de dados consiste numa colecção de objectos de sistema e de utilizadores, armazenados fisicamente em discos. O objectivo do atacante é geralmente o de aceder à base de dados, sendo que a instância é o que medeia o seu acesso (ver Figura 2).

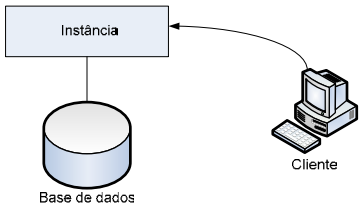

Figura 2 – Instância e base de dados

Os *objectos* são componentes lógicos representados dentro da base de dados com uma determinada estrutura, distinta consoante a tipologia do objecto. Destacamos os seguintes tipos de objectos. Um *índice* é um objecto que contém uma entrada para cada valor que apareça na coluna indexada de uma tabela. Um *stored procedure* contém código T-SQL armazenado na base de dados. Uma *tabela* é o objecto central de qualquer base de dados, cumprindo o propósito de armazenamento de dados. Um *trigger* é um objecto *stored procedure* ligado a uma tabela que é executado após a ocorrência de um evento específico. Uma *vista* é o objecto que representa os dados contidos em uma ou mais tabelas.

A extensão das funcionalidades do SQL (do Inglês *Structured Query Language*) é realizada através da inclusão no motor de base de dados da linguagem T-SQL, que permite a definição de variáveis, condições e lógica de controlo através da utilização de construtores como If…Else, Case, While, etc.

Autenticação O *SQL Server* suporta dois tipos de *autenticação*:

- *Nativa:* o utilizador e palavra-chave são enviados para o *SQL Server* que irá fazer a autenticação localmente na base de dados;
- *Windows:* integrada com o sistema operativo; de forma à autenticação ter sucesso, o utilizador ou grupo terão que existir no servidor ou domínio e terem acesso ao *SQL Server*.

Por omissão, o utilizador que existe na base de dados é o utilizador "sa" com o qual poderão ser desempenhadas todas as tarefas de administração do repositório e da instância.

Autorização O acesso a objectos da base de dados é controlado por *server roles*, *database roles*, privilégios sobre instruções SQL e privilégios sobre objectos. Os *server roles* e *database roles* controlam os comandos que podem ser executados, os dados que podem ser lidos e/ou alterados, e os objectos que podem ser alterados, apagados e/ou criados. Destacam-se os seguintes *server roles* [9]:

- *SYSADMIN*: atribui ao utilizador ou grupo controlo completo sobre o *SQL Server*, as suas bases de dados e todos os seus objectos; por omissão o utilizador "sa" e o grupo "BUILTIN\Administrators" são membros deste *role*;
- *SERVERADMIN*: atribui ao utilizador ou grupo a capacidade de modificar parâmetros do *SQL Server* como quantidade de memória e CPU, etc.

Destacam-se os seguintes *database roles* [9]:

- *DB_ACCESSADMIN*: atribui ao utilizador ou grupo, a capacidade de atribuir ou eliminar acesso à base de dados;
- *DB_OWNER*: atribui ao utilizador ou grupo, controlo total sobre todos os objectos e operações da base de dados; por omissão, o utilizador "dbo" pertence a este *role*;
- *PUBLIC*: todos os utilizadores e grupos pertencem a este *role*.

O *SQL Server* também permite a criação de *user defined roles*, permitindo assim agrupar utilizadores que tenham as mesmas necessidades de privilégios. Os privilégios sobre instruções SQL permitem atribuir a um utilizador operações transversais à base de dados. Destacamos os seguintes privilégios sobre instruções SQL:

- *CREATE FUNCTION*: possibilita a criação de funções T-SQL;
- *CREATE PROCEDURE*: possibilita a criação de procedimentos T-SQL;
- *CREATE TABLE*: possibilita a criação de tabelas.

Os privilégios sobre tabelas são mais granulares que os *roles*, permitindo a atribuição a grupos e/ou utilizadores, ajudando a definir os comandos DML que podem ser usados para acesso a objectos como tabelas, vistas e *stored procedures*.

Auditoria O processo de auditoria de eventos pode ser executado através de um conjunto de métodos que variam essencialmente ao nível de detalhe e das ferramentas utilizadas. Os métodos são os seguintes:

- Auditoria de eventos do *Windows*: os eventos relacionados com o *SQL Server* são armazenados no ficheiro "AppEvent.evt" (*application log*). Os eventos que são auditados são: *startup* e *shutdown* da instância; *backups* e *restores* de bases de dados; alterações de configuração da instância.
- Auditoria do processo de *login*: podem ser auditados todos os *logins*, apenas os que tiveram sucesso ou os que falharam;
- Auditoria C2[2]: é auditada a criação e o acesso a objectos [25].

O método usado por omissão no *SQL Server* origina um ficheiro "ErrorLog" que pode ser analisado, mas que essencialmente contém a mesma informação que o *application log*.

Comunicações O protocolo utilizado na comunicação cliente/servidor é o protocolo proprietário TDS (*Tabular Data Stream*), normalmente usado sobre a pilha TCP/IP, embora também possa ser usado com *named pipes* e com outros mecanismos de comunicação entre processos (ver Figura 3). Por omissão as comunicações são realizadas em claro, embora a ligação cliente/servidor possa ser configurada para ser protegida com SSL/TLS. O processo principal do *SQL Server* aceita por omissão pedidos nos portos TCP/1433 e UDP/1434, sendo que o primeiro é utilizado na comunicação cliente/servidor tradicional.

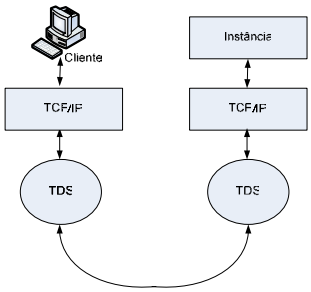

Figura 3 - Comunicações

[2] Primeiro nível da classificação do processo de avaliação definido pelos *Trusted Computer System Evaluation Criteria* (TCSEC) [22], que define um conjunto de eventos a auditar. Para que um sistema possua esta classificação um administrador tem que conseguir auditar com base na identidade de utilizador e objecto.

Pré-avaliação Com base no que ficou dito e numa avaliação documental das duas versões do *SQL Server*, podemos apontar desde já um conjunto de aspectos positivos e negativos da sua segurança por omissão. Aspectos negativos:

- Por omissão as comunicações cliente/servidor são realizadas em claro e os portos TCP e UDP estão no estado LISTEN (*SQL Server 2000*);
- Apesar do processo de auditoria estar ligado, eventos de autenticação não são auditados;
- O utilizador "sa" tem por omissão a palavra-chave vazia (*SQL Server 2000*);
- Um utilizador com perfil de administrador é omnipotente, podendo aceder a quaisquer dados.

Aspectos positivos:

- É suportada a utilização de protocolos seguros como o SSL ou o IPSec nas comunicações cliente/servidor;
- É suportado como mecanismo de autenticação o protocolo Microsoft Kerberos;
- Elevada granularidade no mecanismo de autorização, permitindo o controlo do utilizador, acção e objecto;
- O mecanismo de auditoria possibilita um elevado detalhe de registo, providenciando o registo da acção e do objecto.

Enquadrando os aspectos negativos e os aspectos positivos nas propriedades que definimos como imprescindíveis para um SGBD, apresentamos um quadro resumo da configuração por omissão na tabela 1.

Propriedade	**Configuração por omissão**
Confidencialidade e integridade nas comunicações	Não existe.
Disponibilidade	(Não possuímos informação para responder agora)
Confidencialidade e integridade na informação armazenada	Não existe. (o administrador tem poder total)

Tabela1 – Configuração por omissão em termos de propriedades

Enquadrando os aspectos negativos e os aspectos positivos nos mecanismos que definimos como imprescindíveis para um SGBD, apresentamos um quadro resumo da configuração por omissão na tabela 2.

Mecanismo	**Configuração por omissão**
Autenticação	Fraca, dependente de utilizador e palavra-chave (a qual não necessita de respeitar nenhuma regra).
Autorização	Qualquer utilizador tem autorização para aceder a um conjunto de funcionalidades.
Auditoria	Existe, mas é insuficiente.

Tabela 2 – Configuração por omissão em termos de mecanismos

3.3 Avaliação experimental

As versões utilizadas nos testes – 2000 e 2005 –, possuem pontos de entrada comuns. Não obstante a similaridade arquitectural entre as duas versões, os pontos de entrada embora comuns estão sujeitos a diferentes falhas devido a um natural processo de maturação da segurança do produto. Assim, a secção começa por apresentar as vulnerabilidades encontradas na avaliação experimental comuns a ambas as versões do SGBD, depois incide sobre as vulnerabilidades da versão 2000 e por fim da versão 2005.

3.3.1 SQL Server 2000 e 2005

Ambas as versões apresentam similaridades, de onde destacamos:

- Comunicação TDS entre os clientes e o servidor;
- Existência de um serviço UDP – *SQL Server Resolution Protocol* (SSRP) – criado pela Microsoft para suportar mais do que uma instância no mesmo servidor;
- Existência de um *role* PUBLIC ao qual qualquer utilizador pertence;
- Implementação do mecanismo de *backups*.

Após a instalação por omissão de ambas as versões em máquinas virtuais distintas, utilizámos o utilitário *nmap* para enumerarmos os pontos de entrada possíveis. O resultado encontra-se na tabela 3.

Protocolo	Porto	Serviço
TCP	1433	SQL Server
UDP	1434	SQL Server browser

Tabela 3 – Portos de comunicação encontrados pelo *nmap*.

Fruto do levantamento dos pontos de entrada e da documentação da Microsoft acerca dos mesmos, foram criados dois DFDs. O DFD do *Cliente TDS* tem como objectivo descrever a comunicação cliente/servidor com o SGBD (ver Figura 4). Os aspectos que iremos explorar neste cenário são: segurança do serviço que trata os pedidos TDS dos clientes; segurança na comunicação de um cliente TDS e uma instância do *SQL Server*; capacidade de nos autenticarmos no *SQL Server* recorrendo a credenciais criadas por omissão aquando do processo de instalação.

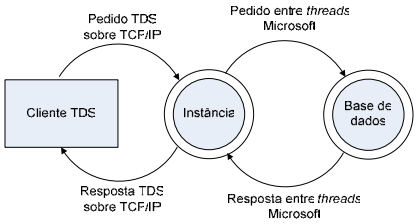

Figura 4 – DFD TDS

O DFD do *cliente SSRP* tem como objectivo comunicar com um servidor *SQL Server*, de forma a obter os nomes das instâncias existentes no servidor (ver Figura 5). Neste cenário iremos explorar a segurança do serviço SSRP.

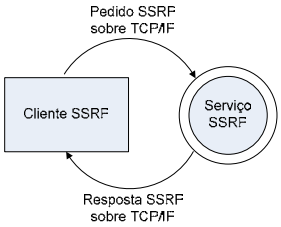

Figura 5 – DFD SSRP

Segurança do serviço que trata os pedidos TDS dos clientes O serviço responsável por permitir o acesso à base de dados por parte de clientes remotos ou locais, utiliza no primeiro caso o porto TCP/1433. Assim, de forma a identificarmos um servidor *SQL Server* (192.168.2.3) podemos usar o utilitário *amap* (a negrito o input):

```
amap 192.168.2.3 1433
Protocol on 192.168.2.3:1433/tcp matches ms-sql
```

Foram efectuados diversos ataques de negação de serviço – SYN flood, TCP connect flood – recorrendo ao utilitário *hping* [9] e a código C desenvolvido, mas nenhum resultou em negação de serviço. No entanto, não foram realizados ataques mais ricos como injecção de pacotes respeitando a gramática TDS ou injecção de pacotes com formatação arbitrária. Uma vez que, por omissão o porto TCP/1433 pode ser acedido a partir de qualquer ponto na rede (bastando o porto TCP estar disponível e permitir o *3-way handshake* do TCP), poderá ser possível a exploração de vulnerabilidades do TCP/IP e/ou do TDS através de ataques similares.

Segurança na comunicação entre um cliente TDS e o SQL Server Utilizando os utilitários *Query Analyzer* e *wireshark*, facilmente comprovamos a ausência de confidencialidade por omissão. Os exemplos abaixo apresentam os resultados obtidos com um servidor *SQL Server 2000*, mas quando usámos um servidor *SQL Server*

2005 os resultados foram similares. No *Query Analyzer*, após nos ligarmos com o utilizador "sa" que tem a palavra-chave em branco, executámos os dois comandos abaixo, obtendo os resultados na tabela 4 – lista das bases de dados configuradas e respectiva localização nos discos internos do servidor.

```
use master;
select * from sysdatabases;
```

Name	dbid	…	Filename
master	1		E:\SQLSRV2000\MSSQL\data\master.mdf
model	3		E:\SQLSRV2000\MSSQL\data\model.mdf
Msdb	4		E:\SQLSRV2000\MSSQL\data\msdb.mdf
northwind	6		E:\SQLSRV2000\MSSQL\data\northwind.mdf
Pubs	5		E:\SQLSRV2000\MSSQL\data\pubs.mdf
tempdb	2		E:\SQLSRV2000\MSSQL\data\tempdb.mdf

Tabela 4 – Resultado de um *select*

O *wireshark* foi usado para escutar a comunicação entre o cliente e o servidor. Um excerto da saída encontra-se no Anexo 1.

A captura foi realizada num cliente onde coexistiam os utilitários *Query Analyzer* e *wireshark*. Para executar um ataque real, o *wireshark* seria executado num cliente distinto, estando o seu sucesso dependente da possibilidade de escutar a comunicação (por exemplo estando o cliente, servidor e o *wireshark* ligados a uma rede Ethernet em difusão).

Ligações ao SGBD Por omissão a autenticação do *SQL Server* é integrada com o sistema operativo, não sendo possível a ligação ao mesmo através de utilizadores que não existam no domínio de autenticação do servidor. Os únicos utilizadores que se podem ligar ao *SQL Server* são os que pertencem ao grupo BULTIN\Administrators. Nesse cenário, teríamos que explorar vulnerabilidades associadas ao sistema operativo o que sai do âmbito deste estudo.

Muitas vezes os administradores de bases de dados alteram o modo de autenticação para *mixed* de forma a poderem usar utilizadores nativos ao *SQL Server* no processo de autenticação, especialmente quando os servidores encontram-se em zonas desmilitarizadas sem acesso a recursos de domínio. No cenário de autenticação *mixed*, os *logins* podem ser criados na base de dados sem correspondência com utilizadores de sistema operativo.

O que podemos fazer com um utilizador vulgar? Criámos através do utilitário *Query Analyzer* um utilizador de nome "utilizador1" com palavra-chave "utilizador1". Apesar de não ter sido dada explicitamente permissão de acesso a qualquer base de dados, o utilizador criado irá "personificar" um utilizador especial denominado "guest" criado pelo *SQL Server* aquando da instalação. Por omissão apenas a base de dados "model" não possui o utilizador "guest". Utilizando o utilitário *FreeTDS*:

Estabelecimento da uma ligação cliente/servidor com o servidor identificado no cliente por SQLSERVER2000, utilizando o utilizador "utilizador1" com palavra-chave "utilizador1":

```
./tsql -S SQLSERVER2000 -U utilizador1 -P utilizador1
```

Execução de uma instrução SQL bem conhecida do *SQL Server*:

```
1> select @@version
2> go
Microsoft SQL Server 2000 - 8.00.2039 (Intel X86)
May 3 2005 23:18:38
Copyright (c) 1988-2003 Microsoft Corporation
Enterprise Edition on Windows NT 5.2 (Build 3790: Service Pack 1)
```

Um atacante ficaria a conhecer a versão exacta do *SQL Server*, como também a versão exacta do sistema operativo.

Consulta do campo "comment" da tabela "sysconfigures" – contém o valor de opções de configuração:

```
1> select value from sysconfigures where comment like '%c2%audit%'
2> go
```

```
value
0
```

O resultado deste comando é útil para um atacante, uma vez que representa o sistema estar ou não configurado para auditoria de nível C2.

Consulta dos campos "name" e "filename" da tabela "sysdatabases" – contém informação acerca das bases de dados existentes no sistema:

```
1> select name, filename from sysdatabases
2> go
name filename
master e:\SQLSRV2000\MSSQL\data\master.mdf
model e:\SQLSRV2000\MSSQL\data\model.mdf
msdb e:\SQLSRV2000\MSSQL\data\msdbdata.mdf
Northwind e:\SQLSRV2000\MSSQL\data\northwnd.mdf
pubs e:\SQLSRV2000\MSSQL\data\pubs.mdf
tempdb e:\SQLSRV2000\MSSQL\data\tempdb.mdf
```

Com o resultado deste comando, um atacante fica a conhecer quais as bases de dados que existem configuradas no servidor e onde se encontram os ficheiros das mesmas.

Consulta dos campos "name", "dbname", "has access" da tabela syslogins – contém informação acerca dos *logins* criados:

```
1> select name, dbname, hasaccess from syslogins
2> go
name dbname hasaccess
BUILTIN\Administrators master 1
sa master 1
utilizador1 master 1
```

Um atacante pode desta forma obter todos os logins existentes, a sua *default database* e se têm ou não acesso à base de dados.

Role PUBLIC Este *role* é criado aquando da criação da base de dados e qualquer utilizador criado na base de dados possui-o. Desta forma, qualquer privilégio do *role* PUBLIC é de facto um privilégio de todos os utilizadores que existam na base de dados. Por omissão existe um conjunto de tabelas que qualquer utilizador pode consultar, de onde destacamos:

- *SYSCOMMENTS*: contém o código de todas as vistas e *stored procedures*;
- *SYSCONFIGURES*: mostra a configuração de um conjunto de parâmetros como o número de ligações que podem existir na base de dados (útil para um ataque de força bruta), entre outros. Também permite a verificação da configuração do processo de auditoria;
- *SYSDATABASES*: mostra todas as bases de dados existentes no servidor;
- *SYSLOGINS*: mostra todos os *logins* existentes.

A este *role* também estão atribuídos privilégios sobre *stored procedures*, de onde destacamos:

- *SP_HELPDB*: mostra informação (nome, espaço utilizado, etc.) acerca de todas as bases de dados existentes no servidor;
- *SP_HELPDEVICE*: mostra informação acerca dos ficheiros constituintes das bases de dados.

Extracção de informação de backups O *SQL Server* suporta cinco tipos de backups: [17]

- *Total*: todos os ficheiros da base de dados são salvaguardados;
- *Transaccional*: apenas os ficheiros de *log* transaccionais são salvaguardados;
- *Diferencial*: são salvaguardadas as diferenças existentes entre os ficheiros contidos no último backup e os existentes;
- *File/Filegroup*: permite que guardemos um *filegroup*;
- *Snapshot*: a capacidade de realizar este tipo de backup que constitui uma réplica de todos os ficheiros e logs transaccionais depende de hardware e de *software* independente.

A realização de um *backup* de base de dados é tradicionalmente efectuada recorrendo à ferramenta *Enterprise Manager* do *SQL Server bultin*, que gera um ficheiro binário por cada base de dados salvaguardada. O problema com estes backups, é que o(s) ficheiro(s) resultantes têm informação não cifrada. Considerando que empiricamente os sistemas utilizados para salvaguardar os backups são máquinas onde a segurança é mais

frágil este facto pode levantar sérios problemas. Por exemplo, obtendo acesso a um destes binários ("master.bak", *backup* da base de dados master) o atacante pode utilizar o comando UNIX *strings* (ver Anexo 2) de forma a obter as cadeias alfanuméricas e assim visualizar código T-SQL de procedimentos existentes na base de dados. Caso o *backup* seja executado remotamente, o tráfego entre o cliente e o servidor está sujeito a ser capturado através do mesmo método que o usado anteriormente para descrever a comunicação cliente/servidor.

Segurança do serviço SSRP O SSRP é o serviço responsável por resolver nomes de instâncias e informar o cliente da forma como se poderá ligar a uma instância em execução num servidor utiliza o porto UDP/1434. Existem vários utilitários que explorando este serviço permitem descobrir os servidores *SQL Server* existentes num segmento de LAN. Um destes utilitários é o *osql* da Microsoft:

```
osql -L
Servers:
ORTHOSIE
```

Outros utilitários como o *SQLPing* ou o *SQLRecon*, ambos criados por Chip Andrews, permitem obter informações valiosas acerca de um servidor, como a sua versão e se a sua configuração é a de um ambiente de alta disponibilidade.

3.3.2 SQL Server 2000

A instalação do *SQL Server 2000* testada foi realizada sobre o sistema operativo Microsoft Windows 2003. A versão instalada foi a última disponível, *SQL Server 2000* com *Service Pack* 4. De acordo com o objectivo, a configuração que resultou do processo de instalação foi a configuração por omissão.

Ligações ao SGBD O *SQL Server 2000* por omissão cria dois logins:

- *BUILTIN\Administrators:* autenticação exterior através do sistema operativo;
- *SA:* utilizado para autenticação local no próprio SGBD com palavra-chave vazia.

Os dois utilizadores possuem o papel de administrador e portanto podem fazer tudo o que quiserem nas bases de dados. Por omissão:

- A autenticação é integrada com o sistema operativo, o que significa que só podemos ter acesso ao *SQL Server* caso tenhamos credenciais de um utilizador do sistema operativo do servidor (ou do domínio, caso seja configurado para tal);
- Nenhum utilizador tem limite quanto ao número de tentativas de login. Isto é particularmente útil para um atacante, pois associado ao facto da auditoria por omissão também não estar ligada, permite fazer ataques *online* à autenticação do *SQL Server*.

Um atacante pode realizar ataques de dicionário, utilizando para o efeito um cliente *SQL Server* (TDS) como o "FreeTDS" (cliente TDS open source) e um programa auxiliar que permita iterar as invocações e ler de um ficheiro de palavras-chave. Exemplo de um automatismo escrito em bash que faz esse ataque:

```
#!/bin/sh
TSQL="/usr/local/freetds/bin/tsql"
USER=sa
DICIONARIO= "/usr/local/freetds/bin/dicionario.txt"
OUTPUT="/usr/local/freetds/bin/output.log"
while read line
do
$TSQL -S SQLSERVER2000 -U $USER -P $line 2> $OUTPUT
INCORRECTO=`cat $OUTPUT | grep incorrect | wc -l`
if [ $INCORRECTO == 0 ]
then
echo "Password descoberta: $line"
exit
fi
done < $DICIONARIO
```

O automatismo percorre o ficheiro "dicionario.txt" gerado previamente e por cada linha executa o TSQL utilizando a palavra-chave lida do ficheiro. A utilização do redireccionamento para *stderr* é o que nos permite aferir se descobrimos ou não a palavra-chave de um utilizador.

Um atacante poderá realizar um ataque de força bruta, necessitando apenas de desenvolver um pequeno programa que crie as sequências a experimentar de acordo com regras que estipule (por exemplo cadeias de 1 a 8 caracteres alfanuméricos), e que por cada sequência gerada invoque o cliente TDS da mesma forma que no ataque de dicionário.

O que é que um atacante pode fazer com um utilizador vulgar? Recorrendo mais uma vez ao utilizador criado anteriormente de nome "utilizador1" e ao utilitário "FreeTDS":

Estabelecimento de uma ligação cliente/servidor com o servidor identificado no cliente por SQLSERVER2000, utilizando o utilizador "utilizador1" com palavra-chave "utilizador1" :

```
./tsql -S SQLSERVER2000 -U utilizador1 -P utilizador1
```

Consulta dos campos "routine_body", "routine_definition" da vista "routines" – contém o código de objectos da base de dados:

```
1> select routine_body, routine_definition from information_schema.routines whe-
re specific_name='sp_helpsort'
2> go
routine_body routine_definition
SQL create procedure sp_helpsort --- 1996/04/08 00:00
AS
set nocount on
/*
** Now display the server default collation name
*/
declare @servercollation sysname
select @servercollation = convert(sysname, serverproperty('collation'))

if @servercollation is not NULL
BEGIN
select 'Server default collation' = description
from ::fn_helpcollations() C
where @servercollation = C.name
END

set nocount off
return(0) -- sp_helpsort
```

A vista "routines" permite obter o DDL de qualquer procedimento armazenado na base de dados. Esta capacidade, conjugada com a de conseguirmos obter o nome de todos os objectos existentes na base de dados, poderá permitir a elaboração de um catálogo de objectos e DDLs interessantes à exploração de vulnerabilidades.

Para além do privilégio de SELECT de um conjunto de tabelas e vistas, ao role PUBLIC também se encontram atribuídos privilégios de EXECUTE sobre alguns procedimentos, de onde destacamos um *extended procedure* designado por XP_DIRTREE.

```
1> exec xp_dirtree 'C:\WINDOWS\SYSTEM32'
2> go
subdirectory depth
1025 1
1028 1
...
1054 1
2052 1
3076 1
3com_dmi 1
administration 1
...
```

Este *extended procedure* permite obter a estrutura de directorias e ficheiros de qualquer unidade lógica, o que permite a um atacante obter informações preciosas acerca do software instalado. Conforme descriminado na documentação da Microsoft, qualquer utilizador pode criar objectos na base de dados "tempdb":

```
CREATE TABLE #teste
(
subdir VARCHAR(255),
profundidade int,
ficheiro int
)
INSERT #teste EXEC xp_dirtree 'c:\',0,1
DECLARE @contador INT
DECLARE @max INT
SET @contador = 1
SET @MAX = 12
WHILE @contador < @MAX
BEGIN
INSERT #teste SELECT * FROM #teste
SET @contador = @contador + 1
END
```

Esta inserção representou um acréscimo de 3.6 Gbytes em 15 minutos, à base de dados "tempdb". Um atacante pode facilmente criar uma tabela contendo múltiplas colunas VARCHAR de forma a maximizar o tamanho de cada registo inserido na tabela e assim rapidamente exaurir o espaço em disco. De forma a controlar o progresso do procedimento T-SQL, o atacante poderá recorrer a um *extended procedure* XP_FIXEDDRIVES que lhe permite monitorizar o espaço disponível em disco – actualizado cada vez que é executado.

```
1> exec xp_fixeddrives
2> go
drive MB free
C 322
E 3695
```

3.3.3 SQL Server 2005

A instalação do *Microsoft SQL Server 2005* foi realizada sobre o sistema operativo *Microsoft Windows 2003*. A versão instalada foi a última disponível, *SQL Server* 2005 com *Service Pack* 1. Por omissão, o *SQL Server* 2005 nas versões *Developer*, *Evaluation* e *Express* não activa a comunicação TCP. Uma vez que pretendemos estudar um cenário em que a comunicação cliente/servidor seja possível, após a instalação do *SQL Server* 2005 foi manualmente activada a comunicação TCP.

Ligações ao SGBD O *SQL Server 2005* por omissão cria dois logins:

- *BULTIN\Administrators:* autenticação exterior através do sistema operativo;
- *SA:* aquando da instalação somos forçados a alterar a palavra-chave deste utilizador.

Esses dois utilizadores possuem o papel de administrador e portanto podem fazer tudo o que quiserem nas bases de dados. Por omissão:

- A autenticação é integrada com o sistema operativo, como na versão 2000;
- Ao contrário do *SQL Server 2000*, existe limite quanto ao número de tentativas de falha no processo de login (imposto pela *local security policy* do sistema operativo) e as tentativas de login não sucedidas são auditadas. Estas alterações impossibilitam ataques *online* à autenticação.

O que é que um atacante pode fazer com um utilizador vulgar? Apesar do acesso a objectos como a vista "ROUTINES" ou o *extended procedure* "XP_DIRTREE", não ser mais possível na versão 2005 do *SQL Server*, existem outros objectos não existentes em 2000 que podem ser utilizados maliciosamente.

Consulta de um conjunto de campos da vista "database_files" – contém informação acerca das bases de dados:

```
1> select type, physical_name, state, size, max_size from sys.database_files
2> go
type physical_name state size max_size
```

```
0 E:\sqlsrv2005\MSSQL.1\MSSQL\DATA\master.mdf 0 512 -1
1 E:\sqlsrv2005\MSSQL.1\MSSQL\DATA\mastlog.ldf 0 160 -1
```

Este objecto permite a um atacante saber por exemplo se os ficheiros que constituem a base de dados têm um limite máximo de tamanho definido. Pode ser interessante para exploração de ataques de negação de serviço.

Consulta de um conjunto de campos da vista "endpoints" – contém informação acerca dos protocolos de comunicação configurados:

```
1> select name, state_desc from sys.endpoints
2> go
name state_desc
TSQL Local Machine STARTED
TSQL Named Pipes STARTED
TSQL Default TCP STARTED
TSQL Default VIA STARTED
```

A informação oferecida por este objecto resume-se ao estado dos protocolos de comunicação existentes no *SQL Server*.

À semelhança do que acontecia no *SQL Server 2000*, também no 2005 qualquer utilizador pode criar objectos na base de dados "TEMPDB". Um atacante pode realizar um ataque de negação de serviço, similar ao efectuado na secção do *SQL Server 2000*, com uma excepção apenas: não pode recorrer ao *extended procedure* "XP_DIRTREE". Mas este facto não é crítico, bastando ao atacante encontrar outra fonte qualquer de informação como por exemplo um "SELECT @@VERSION" adequar o DDL da tabela e definir um máximo adequado para o ciclo de repetição.

3.4 Resultados

Nesta secção apresentamos sumariamente as falhas de segurança que identificámos na secção 3.3. Resumindo e classificando os ataques realizados, quanto ao nível de permissão e/ou capacidade do atacante, apresentamos a seguinte tabela:

Ataque	**Permissão/Capacidade**	**Versão vulnerável**
Confidencialidade nas comunicações	Capturar tráfego cliente/servidor	2000, 2005
Snooping com um utilizador vulgar	Ligação à base de dados com um qualquer utilizador.	2000, 2005
Extracção de informação de *backups*	Capturar tráfego cliente/servidor aquando de um *backup* ou obter acesso ao local de armazenamento	2000, 2005
Ataque de dicionário/força bruta *online*	Acesso ao porto TCP/1433	2000, 2005[3]
Perigo de um utilizador vulgar – negação de serviço (disco)	Ligação à base de dados com um qualquer utilizador	2000, 2005

Tabela 5 – Ataques realizados

Os ataques realizados às instalações por omissão das últimas duas versões do SGBD *Microsoft SQL Server,* permitiram completar a pré-avaliação realizada em 3.2. Enquadrando os aspectos negativos e positivos nas propriedades que definimos como imprescindíveis para um SGBD, apresentamos um quadro resumo da avaliação da instalação por omissão na tabela 6.

Propriedade	**Configuração por omissão**
Confidencialidade e integridade nas comunicações	Não existe.
Disponibilidade	Existe. (considerando os testes realizados)
Confidencialidade e integridade na informação armazenada	Não existe. (o administr. tem poder total)

Tabela 6 – Configuração por omissão em termos de propriedades avaliadas

Enquadrando os aspectos negativos e positivos nos mecanismos que definimos como imprescindíveis para um SGBD, apresentamos um quadro resumo da avaliação da instalação por omissão na tabela 7.

[3] Depende da política de segurança do sistema operativo.

Mecanismo	Configuração por omissão
Autenticação	Fraca, dependente de utilizador e palavra-chave – não existem regras pré-definidas.[4]
Autorização	Qualquer utilizador tem autorização para aceder a um conjunto de funcionalidades.
Auditoria	Está activa, no entanto por omissão não audita eventos de autenticação nem autorização.

Tabela 7 – Configuração por omissão em termos de mecanismos avaliados

Evolução 2005 Foi verificado que a versão 2005 apresenta as seguintes melhorias em relação à versão 2000:

- Por omissão não existem pontos de entrada TCP nem UDP;
- O SGBD herda a política de segurança local do sistema operativo;
- O *role* PUBLIC apresenta privilégios sobre menos objectos potencialmente perigosos.

Uma vez que a instalação por omissão do *SQL Server* 2005 dos testes realizados não introduz nenhuma falha em relação à instalação por omissão do *SQL Server* 2000, concluímos que houve uma evolução significativa na segurança por omissão do SGBD *SQL Server*, o que era de esperar atendendo ao empenho da Microsoft em aplicar a estratégia SD3.

3.5 Configuração cuidada

Consideremos a questão da protecção de um SGBD através da configuração apropriada dos mecanismos de segurança fornecidos pelo fabricante. A instalação por omissão de um SGBD *Microsoft SQL Server* resulta num sistema pouco seguro, não obedecendo a critérios de segurança básicos, conforme demonstrado nas secções anteriores. No entanto, existe variada documentação concebida não só pelo fabricante como também por especialistas em segurança de diferentes afiliações que demonstra como tornar o SGBD mais seguro, recorrendo às funcionalidades que o mesmo concretiza ou pode ser estendido para concretizar. Assim, de seguida enumeramos os problemas encontrados anteriormente e expomos as soluções *Microsoft* (quando existem) que permitem resolver ou reduzir estes mesmos problemas.

Comunicação TDS
P1: Ausência de confidencialidade na comunicação cliente/servidor.
S1: Configuração da cifração de tráfego, através de SSL ou de IPSec [9].

Ligações ao SGBD
P2: Vulnerabilidade a ataque de dicionário ou de força bruta.
S2: Dependente da política de segurança do sistema operativo, deverão ser implementadas: (a) Expiração da palavra-chave, após "d" dias; (b) *Lock* da conta, após "t" tentativas inválidas de acesso; (c) Definição de histórico de palavras-chave, para que não possam ser repetidas as "p" palavras-chave anteriores, aquando da definição de uma nova; (d) Definição da complexidade da palavra-chave, para que as palavras-chave obedeçam a critérios fundamentais como terem um comprimento mínimo, incluírem um carácter numérico, etc; (e) Activar o processo de auditoria, auditando no mínimo os eventos de *log on/off* e acções sem sucesso.

Role PUBLIC
P3: Excesso de privilégios atribuídos a qualquer utilizador criado na base de dados.
S3: O privilégio EXECUTE deve ser retirado.

P4: Capacidade um qualquer utilizador criar uma tabela temporária e inserir registos até ao limite do sistema de ficheiros.
S4: Aparentemente bastaria revogar o privilégio CREATE TABLE, no entanto após o fazermos, a capacidade permanece. Não foi identificada nenhuma solução *Microsoft* para a resolução deste problema.

Extracção de informação de *backups*
P5: Armazenamento da informação em claro.

[4] No caso do *SQL Server* 2005 depende da configuração do sistema operativo

S5: Deverá ser configurada uma palavra-chave de forma a proteger o *backup* através de cifra baseada em palavra-passe (esta é usada para gerar uma chave secreta) [1].

Segurança do serviço SSRP
P6: *SQL Server* 2000 – Descoberta anónima de servidores SQL e suas versões.
S6: Não foi identificada nenhuma solução *Microsoft* para a resolução deste problema.

Outros
P7: *SQL Server* 2000 – A palavra-chave do utilizador "SA" após uma instalação por omissão apresenta-se vazia.
S7: Após a instalação atribuir uma palavra-chave forte.

Comparando a segurança da instalação por omissão, com a segurança após a concretização das soluções supracitadas, identificámos os ataques que continuaram a apresentar uma ameaça, subsistindo ao processo de securização:

- *Snooping com um utilizador vulgar:* apesar de conseguirmos reduzir os privilégios que o *role* PUBLIC possui em ambas as versões do SGBD *SQL Server,* dos testes realizados qualquer utilizador continua a ter um acesso superior ao necessário. A resolução deste problema poderia ser a redução para zero dos privilégios atribuídos ao *role* PUBLIC de forma a seguirmos o princípio do menor privilégio;
- *Capacidade de um qualquer utilizador vulgar criar tabelas temporárias:* não conseguimos revogar o privilégio CREATE TABLE ao *role* PUBLIC nem ao utilizador *guest*. Por omissão nenhum utilizador deveria poder criar objectos temporários de uma forma não regulamentada, i.e., não obedecendo a critérios indexados aos recursos do sistema (utilização de memória, CPU e disco);
- *Descoberta anónima de servidores SQL e suas versões:* esta descoberta é possível devido à existência de um serviço de resolução que tem por objectivo de garantir o suporte a várias instâncias *SQL Server* no mesmo sistema, o qual não autentica os clientes. A resolução deste problema poderia ser exigir autenticação do cliente para fornecer qualquer tipo de informação ou alterar a solução de suporte de várias instâncias, para que todas as ligações a instâncias passem a ser mediadas por uma instância especial, usando o porto de comunicação cliente/servidor tradicional;
- *Omnipotência do Administrador:* não foi identificada nenhuma solução *Microsoft* capaz de limitar um administrador. Um programador poderá recorrer às funcionalidades *built-in* de cifração e decifração de dados para salvaguardar o acesso à informação por parte de utilizadores não autorizados – incluindo o administrador.

4 Comparação entre SGBDs

A secção anterior usou o caso do *Microsoft SQL Server* para apresentar a *metodologia* do estudo reportado neste artigo. Esta secção apresenta os resultados da avaliação da segurança por omissão de todos os SGBDs estudados: *IBM DB2, Microsoft SQL Server, MySQL, Oracle, PostgreSQL* e *Sybase*.

4.1 Configuração por omissão

No preâmbulo de qualquer ataque através de uma ligação cliente/servidor, um atacante procede a um levantamento dos pontos de entrada do alvo com vista à sua enumeração. A tabela 8 apresenta um sumário dos pontos de entrada dos diversos SGBDs estudados.

SGBD	TCP	UDP
IBM DB2 8.2 e 9.1	523, 50000	523
Microsoft SQL Server 2000	1433	1434
Microsoft SQL Server 2005	1433 (activado explicitamente)	
MySQL 4.1 e 5.0	3306	
Oracle 9i	1521, 3340, 7779, 32773	
Oracle 10g	1521, 32798	
PostgreSQL 7.4.14 e 8.2	5432 (activado explicitamente)	
Sybase 12.5 e 15	5000	

Tabela 8 – Pontos de entrada

As configurações por omissão mais seguras são claramente as do *Microsoft SQL Server* 2005 e *PostgreSQL,* uma vez que é necessário activar a comunicação TCP/IP explicitamente.

Ligações ao SGBD Dos SGBDs estudados, todos eles apresentam no final da sua instalação utilizadores criados na própria base de dados ou no sistema operativo do servidor que aloja o SGBD, vide tabela 9.

SGBD	Utilizadores e palavras-chave
IBM DB2 8.2 e 9.1	Criados no sistema operativo (db2inst1, dasusr1, db2fenc1). Aquando da instalação somos obrigados a definir palavras-chave.
Microsoft SQL Server 2000	Existe um utilizador "sa" (administrador) com palavra-chave vazia, no entanto este utilizador por omissão não é usado uma vez que a autenticação é integrada com o *Microsoft Windows*.
Microsoft SQL Server 2005	Aquando da instalação somos forçados a definir a palavra-chave do utilizador "sa" (administrador), embora por omissão ele não seja usado uma vez que a autenticação é integrada com o *Microsoft Windows*.
MySQL 4.1 e 5.0	Existe um utilizador "root" (administrador) com palavra-chave vazia. Por omissão este utilizador apenas pode ser usado localmente.
Oracle 9i	Existem os utilizadores "sys" (administrador), "system" (administrador), "scott" e "dbsnmp". Aquando da instalação somos forçados a alterar as palavras-chave dos utilizadores "sys" e "system". As palavras-chave dos utilizadores "scott" e "dbsnmp" são definidas pelo *Oracle* como "scott" e "dbsnmp" respectivamente.
Oracle 10g	Existem os utilizadores "sys" (administrador), "system" (administrador), "sysman", "dbsnmp" e "mgmt_view". Aquando da instalação somos forçados a alterar as palavras-chave de todos os utilizadores com excepção do "mgmt_view" cuja palavra-chave é gerada aleatoriamente pelo *Oracle*.
PostgreSQL 7.4.14 e 8.2	Existe um utilizador "postgres" (administrador). Por omissão este utilizador apenas pode ser usado localmente. A palavra-chave é definida aquando da criação do utilizador no sistema operativo.
Sybase 12.5 e 15	Existem os utilizadores "sa" (administrador) e "probe". A palavra-chave do "sa" por omissão encontra-se vazia. A palavra-chave do "probe" é gerada pela Sybase.

Tabela 9 – Utilizadores

Não existe uma configuração que possamos indicar como mais segura. O MySQL e o PostgreSQL não permitem acessos remotos sendo necessário activá-los explicitamente, mas a palavra-chave do administrador não obedece a regras – no MySQL encontra-se vazia. Em todos os restantes SGBDs, apenas no caso do *Microsoft SQL Server* 2005 e no *Oracle* 10g é que existe algum tipo de obrigatoriedades na escolha das palavras-chave.

Nenhum dos SGBDs estudados com excepção do *Microsoft SQL Server 2005* – apenas quando instalado sobre *Microsoft Windows 2003 Server* – e *Oracle* 10g implementa por omissão restrições quanto ao número de tentativas de acesso antes do utilizador ser bloqueado. O *Microsoft SQL Server 2005* define ainda uma complexidade da palavra-chave e uma data de expiração da mesma.

Propriedades Do estudo das propriedades que identificámos, nenhum SGBD apresenta uma configuração por omissão satisfatória, ou seja, uma configuração orientada à solução mais segura. Apenas um dos SGBDs estudados apresenta por omissão confidencialidade e integridade nas comunicações, como se pode ver na tabela 10.

SGBD	Confidencialidade e integridade nas comunicações
IBM DB2 8.2 e 9.1	Não existe.
Microsoft SQL Server 2000 e 2005	Não existe.
MySQL 4.1 e 5.0	Não existe.
Oracle 9iR2 e 10gR2	Não existe.
PostgreSQL 7.4.14 e 8.2	Existe. (temos que escolher explicitamente SSL, mas teríamos que escolher alguma coisa)
Sybase 12.5 e 15	Não existe.

Tabela 10 – Confidencialidade e integridade nas comunicações

Apenas dois dos SGBDs resistiram a um ataque simples contra a disponibilidade do serviço de rede. No caso do *Oracle* 10gR2, embora tenha resistido ao ataque realizado, sabemos por experiência própria que não resiste a uma utilização abusiva de uma *probe* TCP da suite HP OpenView, executada com um período de poucos minutos –as suas ligações não são suprimidas automaticamente pelo processo de Oracle. Podemos verificar os dados na tabela 11.

SGBD	Disponibilidade do serviço de rede
IBM DB2 8.2 e 9.1	Ataque simples ao porto TCP/50000 resulta no SGBD ser incapaz de responder a pedidos de novos clientes.
Microsoft SQL Server 2000 e 2005	Ataque realizado não teve sucesso.
MySQL 4.1 e 5.0	Ataque simples ao porto TCP/3306 resulta no SGBD ser incapaz de responder a pedidos de novos clientes.
Oracle 9iR2	Ataque simples ao porto TCP/1521 resulta no SGBD ser incapaz de responder a pedidos de novos clientes.
Oracle 10gR2	Ataque realizado não teve sucesso.
PostgreSQL 7.4.14 e 8.2	Ataque simples ao porto TCP/5432 resulta no SGBD ser incapaz de responder a pedidos de novos clientes.
Sybase 12.5 e 15	Ataque simples ao porto TCP/5000 resulta no SGBD ser incapaz de responder a pedidos de novos clientes.

Tabela 11 – Disponibilidade do serviço de rede

Apenas as duas versões do *MySQL* estudadas e o *Oracle* 10gR2 não permitem que qualquer utilizador crie objectos indiscriminadamente e assim execute um ataque de negação de serviço directo ao(s) sistema(s) de ficheiro(s) e/ou ao sistema de armazenamento lógico do SGBD.

Em relação à confidencialidade e integridade da informação armazenada, nenhum dos SGBDs concretiza por omissão mecanismos que permitam que a informação esteja segura contra leituras ou alterações por parte de um utilizador com perfil de administrador.

Mecanismos Do estudo dos mecanismos que identificámos, nenhum SGBD apresenta uma configuração por omissão satisfatória, ou seja, uma configuração orientada à solução mais segura. Apenas os SGBDs *open source MySQL* e *PostgreSQL* apresentam por omissão um mecanismo de autenticação que classificamos como rico, embora não tenham regras relativamente à palavra-chave. A tabela 12 resume os dados dos diversos SGBDs.

SGBD	Autenticação
IBM DB2 8.2 e 9.1	Pobre, depende de utilizador e palavra-chave – a qual não necessita de obedecer a regras.
Microsoft SQL Server 2000	Pobre, depende de utilizador e palavra-chave – a qual não necessita de obedecer a regras.
Microsoft SQL Server 2005	Pobre, depende de utilizador e palavra-chave – se o sistema operativo for *Microsoft Windows 2003* tem que respeitar regras.
MySQL 4.1 e 5.0	Rica, depende de utilizador, palavra-chave, base de dados e IP do cliente.
Oracle 9iR2 e 10gR2	Pobre, depende de utilizador e palavra-chave – a qual não necessita de obedecer a regras.
PostgreSQL 7.4.14 e 8.2	Rica, depende de utilizador, palavra-chave, base de dados e IP do cliente.
Sybase 12.5 e 15	Pobre, depende de utilizador e palavra-chave – a qual não necessita de obedecer a regras.

Tabela 12 – Autenticação

Apenas o SGBD *MySQL* apresenta por omissão um mecanismo de autorização no qual um utilizador criado na base de dados não herda automaticamente privilégios sobre objectos. No entanto, o *MySQL* permite a um qualquer utilizador a execução de um conjunto de instruções, incluindo uma que pode ser utilizada por um atacante para a execução de um ataque de negação de serviço ao CPU: *benchmark()*. Esta instrução avalia o tempo de execução de um conjunto finito de execuções de outra instrução *MySQL*, permitindo sujeitar o CPU a uma avaliação extremamente extensa decorrente de um pedido absurdo como a medição de 100.000.000 de execuções de uma função criptográfica – *benchmark(100000000, sha1("Esta operacao vai demorar bastante tempo!"))*. Podemos verificar os dados na tabela 13.

SGBD	**Autorização**
IBM DB2 8.2 e 9.1	Qualquer utilizador tem acesso a um conjunto de tabelas que lhe permitem extrair informação acerca do SGBD e BD.
Microsoft SQL Server 2000 e 2005	Qualquer utilizador tem acesso a um conjunto de tabelas, vistas e procedimentos que lhe permitem extrair informação acerca do SGBD e BD.
MySQL 4.1 e 5.0	Não existe acesso a objectos.
Oracle 9iR2 e 10gR2	Qualquer utilizador tem acesso a um conjunto de tabelas, vistas e procedimentos que lhe permitem extrair informação acerca do SGBD, BD e atacar outros sistemas.
PostgreSQL 7.4.14 e 8.2	Qualquer utilizador tem acesso a um conjunto de tabelas que lhe permitem extrair informação acerca do SGBD e BD.
Sybase 12.5 e 15	Qualquer utilizador tem acesso a um conjunto de tabelas, vistas e procedimentos que lhe permitem extrair informação acerca do SGBD e BD.

Tabela 13 – Autorização

Apenas o *Microsoft SQL Server* apresentou uma configuração por omissão com o mecanismo de auditoria ligado. Não obstante, mostra-se insuficiente uma vez que não audita eventos de autenticação nem eventos de autorização. A não configuração do processo de auditoria aliada à ausência de expiração da palavra-chave dos utilizadores e número ilimitado de tentativas de acesso, potencia a realização de ataques de dicionário e/ou força bruta.

4.2 Configuração cuidada

Todos os SGBDs estudados permitem a alteração dos portos TCP e UDP usados por omissão. Embora a segurança oferecida por esta alteração seja relativa, uma vez que continua a ser possível a identificação dos serviços, é bem vinda pois alguns utilitários de ataque automatizados poderão simplesmente não funcionar, reduzindo assim o número de "armas" passíveis de causar problemas.

Ligações ao SGBD Os SGBDs comerciais que estudámos permitem todos eles a imposição de regras quanto à expiração da palavra-chave após "d" dias, bloqueio da conta após "t" tentativas inválidas de acesso e imposição de complexidade às palavras-chave, directa ou indirectamente através do sistema operativo.

Os SGBDs *open source* mostram-se neste campo mais débeis, não oferecendo as mesmas funcionalidades que os comerciais. No entanto são os que graças a um processo de autenticação mais granular – a ligação depende não só de utilizador e palavra-chave, como também de base de dados e IP do cliente – se encontram menos vulneráveis a uma tentativa indiscriminada de acesso.

Propriedades Todos os SGBDs estudados permitem a concretização de confidencialidade e integridade nas comunicações através da utilização de SSL. Alguns, nomeadamente o *IBM DB2* e o *Microsoft SQL Server*, permitem também a utilização de IPSec.

A disponibilidade é uma das propriedades mais sacrificadas. A disponibilidade do serviço de rede de todos os SGBDs com excepção do *Microsoft SQL Server 2000, 2005* e *Oracle* 10gR2 – sabemos que pode ser comprometido com uma *probe* da suite HP OpenView – foi facilmente comprometida através de um ataque de negação de serviço, não tendo sido identificadas soluções dos fabricantes que permitam eliminar esta vulnerabilidade.

A capacidade de qualquer utilizador criar tabelas temporárias é uma funcionalidade que para um atacante possibilita um ataque de negação de serviço. Dos SGBDs vulneráveis – *IBM DB2* 8.2 e 9.1, *Microsoft SQL Server* 2000 e 2005, *Oracle* 9i, *PostgreSQL* 7.4.14 e 8.2, *Sybase* 12.5 e 15 – apenas o *Oracle* 9i apresenta uma solução para o problema.

A confidencialidade e integridade nos dados armazenados são propriedades que podem ser implementados em todos os SGBDs estudados recorrendo às primitivas criptográficas incluídas nos produtos.

Mecanismos Todos os SGBDs estudados permitem a implementação de confidencialidade e integridade no mecanismo de autenticação recorrendo à utilização de SSL. Alguns nomeadamente o *IBM DB2* e o *Microsoft SQL Server* vão mais longe permitindo a utilização de IPSec. Todos os SGBDs com excepção do *MySQL* suportam a utilização do protocolo *Kerberos* no mecanismo de autenticação.

Em relação ao mecanismo de autorização, apesar de podermos limitar o acesso a alguns objectos, revogando privilégios ao *role* PUBLIC e/ou ao próprio utilizador após a sua criação, esta redução mostra-se insuficiente pois após a sua implementação em todos os SGBDs estudados qualquer utilizador permanece com acesso a objectos ou funcionalidades que não deveria ter.

A auditoria é o parente mais pobre dos mecanismos no que diz respeito aos SGBDs *open source*, não existindo de todo. Todos os outros SGBDs permitem a implementação das funcionalidades básicas de auditoria de autenticação e autorização.

4.3 Vulnerabilidades que persistem

As vulnerabilidades para as quais não foram identificadas soluções dos fabricantes encontram-se em sumário na tabela 14.

SGBD	**Vulnerabilidades**
IBM DB2 8.2 e 9.1	Comprometimento anónimo da componente de rede; Comprometimento do espaço em disco; Qualquer utilizador tem privilégios a mais.
Microsoft SQL Server 2000 e 2005	Comprometimento do espaço em disco; Qualquer utilizador tem privilégios a mais.
MySQL 4.1 e 5.0	Comprometimento anónimo da componente de rede; Qualquer utilizador tem privilégios a mais; Não implementa auditoria.
Oracle 9iR2 e 10gR2	Comprometimento anónimo da componente de rede; Qualquer utilizador tem privilégios a mais.
PostgreSQL 7.4.14 e 8.2	Comprometimento anónimo da componente de rede; Comprometimento do espaço em disco; Qualquer utilizador tem privilégios a mais; Não implementa auditoria.
Sybase 12.5 e 15	Comprometimento anónimo da componente de rede; Comprometimento do espaço em disco; Qualquer utilizador tem privilégios a mais.

Tabela 14 – Vulnerabilidades que persistem após configuração

O comprometimento anónimo da componente de rede pode ser mitigado, implementando um processo de controlo cuja única função seja verificar o estado das ligações do serviço de rede e proceder à terminação das sessões sem estado no protocolo por mais que "t" tempo. Neste processo assumimos que os protocolos de sessão usados nos SGBD são *stateful* ou que é possível torná-los *stateful* obrigando à troca de informação de estado entre cliente e servidor. Assumimos ainda que é possível no servidor recorrer à utilização de temporizadores para controlar o tempo de espera por mensagens do cliente – o tempo de espera deveria ser flexível podendo ser estipulado pelo administrador da base de dados de acordo com as necessidades, respeitando o tempo mínimo e máximo permitido na implementação.

A verificação de sem estado no Protocolo poderia ser a verificação de terem sido executados ou não comandos específicos do protocolo durante a sessão ou durante "t" tempo. Neste processo assumimos que os protoco-

los de sessão usados nos SGBD para além de serem *stateful* possuem uma gramática específica com comandos bem conhecidos pelos clientes e servidor, sendo por conseguinte possível validar a conversação e não apenas o envio e recepção de *bytes*.

Para além do processo supracitado, seria importante reduzir os clientes capazes de estabelecer ligações TCP e/ou UDP com os SGBDs para o estritamente necessário. O SGBD *Oracle* apresenta uma funcionalidade denominada por *TCP Valid Node Checking* que pode ser usado para restringir os clientes que se podem ligar ao SGBD. No entanto, tanto quanto sabemos permite à mesma a ligação TCP ao *socket* do *listener* e portanto neste cenário permitiria à mesma o sucesso de um ataque de negação de serviço.

Recorrendo a software exterior ao SGBD, poderemos instalar uma antepara de segurança (*firewall*) – como dispositivo próprio ou instalada no servidor SGBD –, que por omissão bloqueie todas as ligações, permitindo apenas aquelas que configurámos explicitamente.

O comprometimento do espaço em disco verifica-se porque qualquer utilizador pode criar tabelas temporárias. Esta capacidade deveria ser representada por um privilégio específico que teria que ser atribuído explicitamente ao utilizador, como no *Oracle*.

A resolução do problema de excesso de privilégios atribuídos a qualquer utilizador implica uma alteração de fundo no mecanismo de autorização, uma vez que:

- Presentemente caso retiremos todos os privilégios ao *role* PUBLIC existem funcionalidades que simplesmente deixam de funcionar, chegando ao extremo de termos problemas na própria criação de sessão;
- Alguns privilégios não estão atribuídos através do *role* PUBLIC, sendo necessário escolher atribuí-los ou não, garantindo a premissa de que a atribuição final de capacidades ao utilizador deve ser feita com base no princípio do menor privilégio.

4.4 Configuração por omissão orientada à segurança

Os resultados da avaliação de segurança efectuada permitem-nos dizer que a segurança oferecida pela configuração por omissão dos SGBDs não é a adequada.

Tomando como exemplo o *Microsoft SQL Server* 2005, podemos afirmar que embora seja mais seguro que a versão que o antecedeu, persistem configurações fracas nomeadamente no que se refere aos mecanismos de autorização e auditoria. A existência destas falhas na configuração por omissão, é representativa de que não obstante a estratégia SD3+C ter produzido uma versão mais segura do produto, ainda não podemos caracterizar a sua configuração por omissão como uma configuração por omissão segura.

No nosso entender, uma configuração orientada à segurança deverá cumulativamente apresentar as seguintes características:

Comunicações Todas as comunicações não locais com o SGBD deverão ter que ser configuradas explicitamente. Após esta configuração, os utilizadores com papel de administrador deverão continuar a não poder realizar ligações cliente/servidor utilizando protocolos que não locais ao sistema – somos da opinião de que a capacidade um administrador se ligar a um SGBD remoto a ser implementada, deverá obrigar a uma configuração muito específica.

A concretização dos protocolos cliente/servidor deverá avaliar a gramática da comunicação realizada nos *sockets* utilizados pelo SGBD e decidir quanto ao término ou continuação das ligações de forma a reduzir o número de ataques de negação de serviço.

A concretização de um mecanismo de *rate limiting* poderá providenciar a protecção adicional necessária para mitigar ataques de negação de serviço mais evoluídos que respeitem a gramática dos protocolos cliente/servidor [5].

Confidencialidade e integridade nas comunicações Deverá ser assegurada através do encapsulamento do tráfego cliente/servidor no protocolo IPSec – note-se que a degradação de desempenho é cada vez menos importante devido às melhorias significativas na velocidade dos sistemas e redes.

Confidencialidade e integridade na informação Deverá ser assegurada através da cifração de chave pública e da definição de regras e capacidades no acesso a dados aplicacionais – o administrador não pode ser excepção.

Autenticação O mecanismo de autenticação deverá ser rico, recorrendo não só ao par *{utilizador, palavra-chave}* como também ao endereço IP do cliente. A utilização de certificados digitais poderá ser uma solução interessante, mas obriga a recorrer a uma autoridade de certificação, o que pode não ser sempre conveniente.

Autorização Após a instalação não deverão existir pares *{utilizador, palavra-chave}* bem conhecidos e apenas deverão existir os estritamente necessários para o correcto funcionamento do SGBD;

A criação de qualquer utilizador deverá respeitar regras fundamentais quanto à palavra-chave: ser obrigatório um comprimento mínimo (ex. 8 caracteres) e complexidade (ex. 2 caracteres especiais);

A autorização inicial deverá apresentar-se como sendo nula, i.e., um utilizador criado no SGBD não deverá poder aceder a quaisquer tipo de objectos – tabelas, vistas, *stored procedures*, funções, etc. –, não deverá poder executar qualquer tipo de instruções *built-in*, nem tão pouco deverá ter a capacidade de criar objectos temporários.

Auditoria O mecanismo de auditoria deverá registar quaisquer eventos de autenticação e autorização – note-se que o volume da informação é cada vez menos relevante devido ao decréscimo do custo por GB.

Outros Por omissão deverão ser realizados *backups* seguros através da utilização de ferramentas criptográficas *built-in*.

Embora a crescente importância dos SGBD tenha originado a inclusão no motor de base de dados de um conjunto diversificado de tecnologias como o tratamento de informação espacial, tipos XML nativos, linguagens orientadas a objectos, etc., o estudo realizado permite concluir que os fabricantes precisam de conciliar as novas funcionalidades com o incremento da segurança por omissão.

5 Conclusão

Este artigo apresentou os resultados de um estudo da segurança na configuração por omissão de um conjunto de SGBDs – *IBM DB2, Microsoft SQL Server, Oracle, MySQL, PostgreSQL, Sybase* –, detalhando o caso da análise das duas últimas versões do *Microsoft SQL Server*.

Do estudo comparativo, concluímos que existe ainda um longo caminho a percorrer para caracterizarmos qualquer um dos SGBDs como detentores de uma configuração por omissão orientada à segurança, uma vez que foram identificadas falhas nucleares em todos eles. Da análise do *SQL Server*, concluímos que o *Microsoft SQL Server* 2005 é mais seguro que o seu antecessor, apresentando claros benefícios da introdução da estratégia SD3+C no ciclo de desenvolvimento do produto.

A análise efectuada carece de uma quantificação que permita representar numericamente a segurança por omissão e assim comparar algebricamente a segurança por omissão do SGBD A com a do SGBD B. Acreditamos que uma forma de o fazer será através da adopção de um método similar ao desenvolvido por Howard *et al.* [7], onde é proposta uma métrica para determinar se uma versão de um sistema é mais segura do que outra.

Referências

[1] S. Christman, J. Hayes, "Guide to the Secure Configuration and Administration of Microsoft SQL Server 2000", Report Number: C4-50R-02, Version 1.5, National Security Agency, August 2003.

[2] T. Gallagher, B. Jeffries, L. Landauer, "Hunting Security Bugs", Microsoft Press, 2006.

[3] L. A. Gordon and M. P. Loeb and W. Lucyshyn and R. Richardson, "2006 CSI/FBI Computer Crime and Security Survey", Computer Security Institute, 2006.

[4] C. Graham, “Market Share: Relational Database Management Systems by Operating System, Worldwide, 2005”, Gartner Group, May 2006.
[5] A. Householder, L. Manion, G. M. Pesante, R. Weaver and R. Thomas, “Managing the Threat of Denial-of-Service Attacks”, CERT Coordination Center, October 2001.
[6] M. Howard, D. LeBlanc, “Writing Secure Code”, 2ª edição, Microsoft Press, 2003
[7] M. Howard, J. Pincus, J. M. Wing, “Measuring Relative Attack Surfaces”, Chapter 8, in Computer Security in the 21st Century , D.T. Lee, S.P. Shieh, and J.D. Tygar, editors, pp. 109-137, Springer, March 2005.
[8] M. Howard, “Thinking About Security: Secure by Design, Secure by Default, Secure in Deployment and Communication”, http://msdn.microsoft.com/msdntv/transcripts/20030513SecurityMHTranscript.aspx
[9] hping, http://www.hping.org
[10] M. Lewis, “SQL Server Security Distilled”, 2ª edição, Apress, 2004.
[11] S. Lipner, M. Howard, “The Trustworthy Computing Security Development Lifecycle”, Microsoft Developer Network, March 2005, http://msdn2.microsoft.com/en-us/library/ms995349.aspx
[12] D. Litchfield, C. Anley, J. Heasman, B. Grindlay, “The Database Hacker’s Handbook: Defending Database Servers”, Wiley Publishing, 2005.
[13] H. Moniz,, “Oracle Database: a Security Perspective”, Manuscrito não publicado, Faculdade de Ciências da Universidade de Lisboa, January 2006.
[14] National Computer Security Center, “A Guide to Understanding Audit in Trusted Systems”, NCSC-TG-01, Version 2, June 1988.
[15] S. Redwine, N. Davis, Editors, “Processes to Produce Secure Software”, Software Process Subgroup of the Task Force on Security across the Software Development Lifecycle, National Cyber Security Summit, March 2004.
[16] J. H. Saltzer, M. D. Schroeder, “The Protection of Information in Computer Systems”, Proceedings of the IEEE, 63-9, pages 1278-1308, September 1975.
[17] J. R. Shapiro, “Microsoft SQL Server 2005 The Complete Reference”, 2ª edição, McGraw-Hill, 2006.
[18] D. Turner and S. Entwisle and O. Friedrichs and D. Ahmad and J. Blackbird and M. Fossi and D. Hanson and S. Gordon and D. Cole and D. Cowlings and D. Morss and B. Bradley and P. Szor and E. Chien and J. Ward and J. Gough and J. Talbot, “Symantec Internet Security Threat Report. Trends for January 05-June 05”, Volume VIII, Symantec, September 2005.
[19] S. Vaughan-Nichols, “OpenBSD: The most secure OS around”, ZDNet, November 2001, http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2822483,00.html
[20] Checkpwd, http://www.red-database-security.com/software/checkpwd.html
[21] EMS MySQL, http://www.sqlmanager.net/products/mysql/manager
[22] EMS SQL Query for DB2, http://www.sqlmanager.net/en/products/db2/query
[23] FreeTDS, http://www.freetds.org
[24] Microsoft osql, http://technet.microsoft.com/en-us/library/aa213088(SQL.80).aspx
[25] Microsoft SQL Server 2000 C2 Evaluation http://www.microsoft.com/technet/security/prodtech/sqlserver/sqlc2.mspx
[26] Nikto, http://www.cirt.net/code/nikto.shtml
[27] nmap Security Scanner, http://insecure.org/nmap/
[28] Oracle SQL*Plus, http://www.oracle.com/technology/tech/sql_plus/
[29] pgAdmin III PostgreSQL Tools, http://www.pgadmin.org
[30] THC-Amap, http://freeworld.thc.org/thc-amap/
[31] Tnscmd, http://www.jammed.com/~jwa/hacks/security/tnscmd/tnscmd-doc.html
[32] Wireshark, http://www.wireshark.org/

Anexos

Anexo 1 – Excerto da saída

```
0030 40 92 dc b3 00 00 01 01 00 5a 00 00 01 00 75 00 @....... .Z....u.
0040 73 00 65 00 20 00 6d 00 61 00 73 00 74 00 65 00 s.e. .m. a.s.t.e.
0050 72 00 3b 00 0d 00 0a 00 73 00 65 00 6c 00 65 00 r.;..... s.e.l.e.
0060 63 00 74 00 20 00 2a 00 20 00 66 00 72 00 6f 00 c.t. .*. .f.r.o.
0070 6d 00 20 00 73 00 79 00 73 00 64 00 61 00 74 00 m. .s.y. s.d.a.t.
0080 61 00 62 00 61 00 73 00 65 00 73 00 0d 00 0a 00 a.b.a.s. e.s.....
```

A instrução SQL escrita é enviada em claro pelo cliente

```
...
0340 12 00 4e 00 6f 00 72 00 74 00 68 00 77 00 69 00 ..N.o.r. t.h.w.i.
0350 6e 00 64 00 06 00 01 00 01 00 00 1c 00 00 00 00 n.d..... ........
0360 00 00 41 86 8f 00 00 ed bd 1b 00 00 00 00 00 00 ..A..... ........
```

```
0370 00 00 00 00 00 00 00 50 4a 00 65 00 3a 00 5c 00 .......P J.e.:.\.
0380 53 00 51 00 4c 00 53 00 52 00 56 00 32 00 30 00 S.Q.L.S. R.V.2.0.
0390 30 00 30 00 5c 00 4d 00 53 00 53 00 51 00 4c 00 0.0.\.M. S.S.Q.L.
03a0 5c 00 64 00 61 00 74 00 61 00 5c 00 6e 00 6f 00 \.d.a.t. a.\.n.o.
03b0 72 00 74 00 68 00 77 00 6e 00 64 00 2e 00 6d 00 r.t.h.w. n.d...m.
03c0 64 00 66 00 02 1b 02 d1 08 00 70 00 75 00 62 00 d.f..... ..p.u.b.
03d0 73 00 05 00 01 00 01 00 00 18 00 00 00 00 00 00 s....... ........
03e0 41 86 8f 00 00 e0 bb 1b 00 00 00 00 00 00 00 00 A....... ........
03f0 00 00 00 00 00 50 42 00 65 00 3a 00 5c 00 53 00 .....PB. e.:.\.S.
0400 51 00 4c 00 53 00 52 00 56 00 32 00 30 00 30 00 Q.L.S.R. V.2.0.0.
0410 30 00 5c 00 4d 00 53 00 53 00 51 00 4c 00 5c 00 0.\.M.S. S.Q.L.\.
0420 64 00 61 00 74 00 61 00 5c 00 70 00 75 00 62 00 d.a.t.a. \.p.u.b.
0430 73 00 2e 00 6d 00 64 00 66 00 02 1b 02 d1 0c 00 s...m.d. f.......
0440 74 00 65 00 6d 00 70 00 64 00 62 00 02 00 01 00 t.e.m.p. d.b.....
0450 01 00 00 08 00 00 00 00 00 00 41 25 99 00 00 ee ........ ..A%....
0460 2d 46 01 00 00 00 00 00 00 00 00 00 00 00 50 -F...... .......P
0470 46 00 65 00 3a 00 5c 00 53 00 51 00 4c 00 53 00 F.e.:.\. S.Q.L.S.
0480 52 00 56 00 32 00 30 00 30 00 30 00 5c 00 4d 00 R.V.2.0. 0.0.\.M.
0490 53 00 53 00 51 00 4c 00 5c 00 64 00 61 00 74 00 S.S.Q.L. \.d.a.t.
04a0 61 00 5c 00 74 00 65 00 6d 00 70 00 64 00 62 00 a.\.t.e. m.p.d.b.
04b0 2e 00 6d 00 64 00 66 00 02 1b 02 fd 10 00 c1 00 ..m.d.f. ........
04c0 06 00 00 00 ....
```

A resposta à execução da instrução SQL é enviada em claro pelo servidor

Anexo 2 – *Strings*

```
[teste@linux teste]$ strings master.bak | grep -i “create”
/* Procedure for 8.0 server */
CREATE PROCEDURE sp_sproc_columns (
...
if @column_name is null /* If column name not supplied, match all */
select @column_name = '%'
if @procedure_qualifier is not null
begin
if db_name() <> @procedure_qualifier
begin
if @procedure_qualifier = ''
begin
/* in this case, we need to return an empty result set */
/* because the user has requested a database with an empty name */
select @procedure_name = ''
select @procedure_owner = ''
end
else
begin /* If qualifier doesn't match current database */
raiserror (15250, -1,-1)
return
...
```