infra
MAGAZINE

Edição 02 :: Ano I

DevMedia

ESPECIAL: GNU/Linux
Obtendo alta disponibilidade com soluções open source

Política de Segurança da Informação e Firewall
Uma integração necessária para a organização

Administrando sistemas remotamente
Acesso remoto utilizando SSH e Port Knocking

WINDOWS AZURE

Um estudo sobre computação nas nuvens

Desafios da telefonia IP
Diferenças entre telefonia IP e voz sobre IP

Os milagres do Jail no FreeBSD
Podemos usar o Jail em um ambiente na nuvem?

Segurança de ativos
Conheça a Norma ISO 27002

RENOVE JÁ!

Sua assinatura pode estar acabando



Renovando a assinatura de sua revista favorita você ganha brindes e descontos exclusivos.

Faça a renovação de sua assinatura agora mesmo.



www.devmedia.com.br/renovacao ou (21)3382-5038

DevMedia

Sumário

Olá, eu sou o **DevMan**! Desta página em diante, eu estarei lhe ajudando a compreender com ainda mais facilidade o conteúdo desta edição. Será um prazer contar com sua companhia! Confira abaixo o que teremos nesta revista:



Segurança, Servidores

06 – Segurança física dos ativos
Conhecendo a Norma ISO 27002
[*Roberto Henrique*]

Tutorial, Web, Vanguarda

12 – Cloud Computing com o Windows Azure
Um estudo sobre computação em nuvem
[*Giuliano Moreno, Renato Groffe e Tiago Valim*]

Core, Tutorial

28 – Os milagres do Jail no FreeBSD
Podemos usar o Jail em um ambiente na nuvem?
[*Denis Augusto*]

Core, Vanguarda

35 – Desafios da telefonia IP
Conheça as diferenças entre a telefonia IP e a voz sobre IP
[*André Koide da Silva*]

Tutorial, Boas Práticas, Sistemas Operacionais

41 – Alta disponibilidade com GNU/Linux
Implemente ambientes de alta disponibilidade utilizando soluções open source
[*Mário de Mello Bittencourt Neto*]

Core, Segurança, Servidores

49 – Administrando sistemas remotamente
Acesso Remoto com Segurança usando SSH e Port Knocking
[*Flávio Alexandre dos Reis e Eduardo Pagani Julio*]

Boas Práticas, Segurança

58 – Política de Segurança da Informação e Firewall
Uma integração necessária para a organização
[*Paulo Moreira*]

[Servidores] Destaque para itens como instalação, configuração e gerência, de servidores DHCP, DNS, Web, etc.

[Sistemas Operacionais] Foco em características de sistemas operacionais como Windows, Linux, FreeBSD, entre outros.

[Segurança] Artigo dentro do contexto de Segurança: redes, servidores, sistemas operacionais, infraestrutura em geral.

[Core] Técnicas, ferramentas, e outros assuntos que fogem ás demais categorias.

[Tutorial] Artigo no estilo tutorial passo a passo.

[Web] Artigos sobre ou que envolvam tecnologias de infraestrutura para WEB.

[Vanguarda] Artigos sobre tecnologias de ponta, que ainda não fazem parte do dia a dia do administrador mas que prometem ser a próxima febre.

[Boas Práticas] Um dos objetivos da revista é levar para o leitor não somente as melhores técnicas, mas também as melhores práticas, orientadas ao aumento da qualidade e produtividade.

infra MAGAZINE

Ano I • Edição 02 • 2011

Edição

Editor

Eduardo Spínola (*eduspinola@gmail.com*)

Produção

Jornalista Responsável Kaline Dolabella - JP24185

Atendimento ao leitor

A DevMedia possui uma Central de Atendimento on-line, onde você pode tirar suas dúvidas sobre serviços, enviar críticas e sugestões e falar com um de nossos atendentes. Através da nossa central também é possível alterar dados cadastrais, consultar o status de assinaturas e conferir a data de envio de suas revistas. Acesse **www.devmedia.com.br/central**, ou se preferir entre em contato conosco através do telefone 21 3382-5038.

Publicidade

publicidade@devmedia.com.br – 21 3382-5038

Anúncios – Anunciando nas publicações e nos sites do Grupo DevMedia, você divulga sua marca ou produto para mais de 100 mil desenvolvedores de todo o Brasil, em mais de 200 cidades. Solicite nossos Media Kits, com detalhes sobre preços e formatos de anúncios.

Fale com o Editor!

É muito importante para a equipe saber o que você está achando da revista: que tipo de artigo você gostaria de ler, que artigo você mais gostou e qual artigo você menos gostou. Fique a vontade para entrar em contato com os editores e dar a sua sugestão!

Se você estiver interessado em publicar um artigo na revista ou no site Easy Java Magazine, entre em contato com o editor, informando o título e mini-resumo do tema que você gostaria de publicar:

Eduardo Spínola - Editor da Revista
eduspinola@gmail.com

DevMedia

Editorial

Neste momento, gostaria de começar agradecendo pelo imenso apoio dado por vocês à primeira edição da Infra Magazine. Sempre nos planejamos para fornecer o melhor material possível, mas mesmo assim, por melhor que fossem nossas metas, não esperávamos ser tão bem acolhidos neste tão importante espaço de Infraestrutura e Tecnologia da Informação. Todos os feedbacks foram bastante positivos e possibilitarão o planejamento de edições cada vez melhores. Muito obrigado a todos vocês!

Agora, vamos conhecer os artigos da segunda edição! Como matéria de capa, **Cloud Computing com o Windows Azure**, apresenta as principais características do padrão computacional conhecido como cloud computing (ou Computação em Nuvem). Em seguida, será apresentada a iniciativa disponibilizada pela Microsoft dentro deste segmento: a plataforma Windows Azure. Com ela, pode-se alcançar maior flexibilidade para as operações de TI, assim como a redução de custos com pessoal especializado, hardware e software.

O artigo **Os milagres do Jail no FreeBSD e os desafios da Virtualização** aborda como construir uma solução segura que eleve a proteção de um projeto que tem como meta publicar um servidor para acessos de qualquer usuário na Internet. Para isso, será utilizado o Jail como base de um processo de virtualização.

Um tema que até hoje é negligenciado por empresas brasileiras é a **Segurança física dos ativos**. Neste contexto, ainda há muito que se fazer, principalmente no âmbito das pequenas e médias empresas, que possuem grandes dificuldades em adotar requisitos básicos de segurança. Assim, conheceremos a norma ISO 27002, que tem como objetivo auxiliar a implantação, a manutenção e a melhoria contínua dos controles de segurança da informação.

As tecnologias de voz sobre IP estão em crescente expansão e popularização. Deste modo, **Desafios da telefonia IP** procura esclarecer as diferenças, características e vantagens de sua utilização e implementação. Com isso, os profissionais e demais usuários da área de Tecnologia da Informação poderão compreender melhor como integrar e extrair os benefícios da adoção destas soluções em suas empresas.

Em **Política de Segurança da Informação e Firewall** veremos a importância de integrar um Firewall com uma Política de Segurança da Informação. Para isso, será explicado o que é, qual a funcionalidade e a importância de cada um. Uma Política de Segurança da Informação bem definida é a base para garantir o correto uso da informação.

Administrando sistemas remotamente tem como objetivo demonstrar de forma prática uma técnica capaz de confundir scanners mal intencionados na procura por serviços de acesso remoto ativos no servidor, tornando a porta de acesso remoto do SSH oculta para estas ameaças. A técnica de port knocking, aliada às configurações de segurança do SSH, cria um canal confiável de comunicação para servidores GNU/Linux.

Para encerrar esta edição, **Alta disponibilidade com GNU/Linux** mostra como implementar um ambiente de alta disponibilidade utilizando apenas soluções open source. Este assunto é de grande importância para sabermos como manter um serviço que não pode parar e/ou tenha um alto número de acessos e seja necessário minimizar os impactos que uma manutenção ou problema de hardware venha causar.

Seja bem-vindo à Infra Magazine!

Boa leitura e até a próxima!

Eduardo Oliveira Spínola
eduspinola@gmail.com
twitter.com/Java_Magazine



Dificuldades em encontrar conteúdo sobre .Net para iniciantes?



A DevMedia simplifica pra você!

easy .net magazine

Easy .Net Magazine:

A primeira revista digital feita especialmente para quem está dando os primeiros passos em .Net!

Você encontrará exemplos e modelos fáceis e práticos, noções introdutórias e conceitos básicos.

Conheça as vantagens:

- Vídeos e apresentações "embutidos" nas páginas;
- Ilustrações, diagramas e modelos apresentados em PowerPoint;
- Entrevistas em vídeo;
- Códigos-fonte que podem ser copiados com CTRL+C/CTRL+V.

A Easy.Net não é feita em PDF ou Flash!
Ela utiliza um aplicativo 100% online.



DevMedia group

Acesse agora a edição Nº 1 totalmente grátis e comprove

www.devmedia.com.br/easynet

Easy.Net Magazine, seu curso inicial de .Net!

Segurança física dos ativos

Conhecendo a Norma ISO 27002

Roberto Henrique

Nunca se falou tanto em Segurança da Informação como no último ano. Foram diversos eventos, artigos, reportagens, palestras e lançamentos de novos produtos e serviços ao longo de 2010 que contribuíram para o fortalecimento deste setor. Porém, no dia a dia das empresas ainda há muito que se fazer, principalmente no âmbito das pequenas e médias empresas (PMEs), que ainda possuem grandes dificuldades em adotar requisitos básicos de segurança, sobretudo na infraestrutura que suporta o negócio da empresa.

Enxergar o departamento de TI como uma área que suporta toda a operação relacionada ao negócio da empresa ainda é algo raro no meio empresarial das PMEs. Se mal há investimentos em equipamentos e sistemas básicos para o funcionamento da rede, é de se esperar que não haja qualquer melhoria em questões de segurança. Parte da causa deste problema, em muitos casos, é do próprio gestor do departamento de TI, que por ter uma formação puramente técnica, acaba tendo dificuldades em apresentar argumentos na hora de buscar investimentos para sua área.

Em outros casos, os argumentos são mesmo difíceis de mensurar, principalmente investimentos em segurança que muitas vezes só têm um retorno palpável para o empresário quando ele percebe que a ocorrência de uma falha não paralisou suas atividades. Outros tipos de investimento só serão notados ao longo do tempo por não se tratar de um produto físico, mas de uma mudança cultural, que envolve treinamento e conscientização de toda a equipe.

Mas no que se diz respeito à segurança da infraestrutura, há diversos investimentos que podemos dizer que são um pouco mais "fáceis" de justificar, desde que os argumentos mostrem um ganho ao negócio. É o que chamamos de Entrega de Valor ao Cliente.

Quando falamos de segurança de uma infraestrutura de rede, pensamos em datacenters robustos, verdadeiras salas-cofre, rede elétrica estabilizada com potentes no-breaks e geradores de grande porte, além de robôs automatizados de backup. Mas se avaliarmos de forma mais detalhada, veremos que há outras inúmeras ações a serem estudadas para garantir um nível adequado de segurança da infraestrutura de rede. Todas estas ações estão organizadas na Norma ABNT NBR ISO/IEC 27002:2005 - *Código de Prática para a Gestão de Segurança da Informação.*

Esta norma tem como objetivo auxiliar a implantação, a manutenção e a melhoria contínua dos controles de segurança da informação, padronizando diretrizes e procedimentos que auxiliarão a organização na sua gestão. A Norma ISO 27002 possui 133 controles, divididos em 11 seções, que abrangem a segurança da

Resumo DevMan

De que se trata o artigo:

Este artigo discorre sobre a segurança física dos ativos. Para isso, tem seu foco direcionado para a explicação da seção 9 da norma ISO 27002. Esta norma tem como objetivo auxiliar a implantação, a manutenção e a melhoria contínua dos controles de segurança da informação, padronizando diretrizes e procedimentos que auxiliarão a organização na sua gestão. A Norma ISO 27002 possui 133 controles, divididos em 11 seções, que abrangem a segurança da informação em todos os seus aspectos, tratando de ferramentas, processos e pessoas.

Para que serve:

No dia a dia das empresas, ainda há muito que se fazer quando o assunto é segurança física dos ativos. Principalmente no âmbito das pequenas e médias empresas, que ainda possuem grandes dificuldades em adotar requisitos básicos de segurança, sobretudo na infraestrutura que suporta o negócio da empresa.

Em que situação o tema é útil:

Contribuir para a proteção dos ativos que representam valor para a organização, incluindo equipamentos e recursos humanos.



informação em todos os seus aspectos, tratando de ferramentas, processos e pessoas.

Além da proteção física dos ativos, a norma apresenta uma série de recomendações que visam proteger a informação em três fundamentais aspectos:

• **Confidencialidade:** este aspecto da informação visa garantir que somente as pessoas previamente autorizadas tenham acesso à informação. Neste caso, por exemplo, utilizamos sistemas com contas de acesso exclusivas, senhas individuais e softwares de criptografia para proteger o acesso e a comunicação dos dados;

• **Integridade:** garantir que a informação permaneça autentica e que se for alterada seja somente por pessoas autorizadas. A integridade é importante principalmente para garantir a autenticidade de informações disponibilizadas publicamente. Este tipo de informação deve ser preservado para que ninguém altere seu conteúdo ou seu autor;

• **Disponibilidade:** garantir que a informação esteja disponível sempre que for necessário para as pessoas autorizadas. Para atingir este objetivo é que utilizamos sistemas de backup, links redundantes, servidores de contingência, por exemplo.

Não necessariamente uma informação deve possuir estes três aspectos para garantir sua proteção. De acordo com a classificação da informação é que o proprietário poderá definir o nível adequado de segurança.

Para garantir estes três fundamentais aspectos da segurança da informação, a Norma ISO 27002 abrange em suas 11 seções uma série de ações, que envolvem soluções tecnológicas, documentação de processos e conscientização de pessoas.

A seguir, temos um resumo da Norma (itens numerados a partir de 5):

5. **Política da Segurança da Informação:** prover uma orientação e apoio da direção para a segurança da informação de acordo com os requisitos do negócio e com as leis e regulamentações pertinentes;

6. **Organizando a Segurança da Informação:** estrutura de gerenciamento para iniciar e controlar a implementação da segurança da informação na organização;

7. **Gestão de Ativos:** alcançar e manter a proteção adequada dos ativos da organização;

8. **Segurança em Recursos Humanos:** assegurar que os funcionários, fornecedores e terceiros entendam suas responsabilidades e estejam de acordo com os seus papéis, e reduzir o risco de furto, roubo, fraude ou mal uso de recursos;

9. **Segurança Física e do Ambiente:** prevenir o acesso físico não autorizado, danos e interferências com as instalações e informações da organização;

10. **Gestão das Operações e Comunicações:** garantir a operação segura e correta dos recursos de processamento da informação;

11. **Controle de Acesso:** controlar o acesso à informação com base nos requisitos de negócio e segurança da informação;

12. **Aquisição, Desenvolvimento e Manutenção de Sistemas de Informação:** garantir que a segurança seja parte integrante de sistemas da informação;

13. **Gestão de Incidentes de Segurança da Informação:** assegurar que fragilidades e eventos de segurança da informação associados com sistemas de informação sejam comunicados, permitindo a tomada de ação em tempo hábil;

14. **Gestão da Continuidade do Negócio:** não permitir a interrupção das atividades dos negócios e proteger os processos críticos contra efeito de falhas ou desastres significativos, e assegurar a sua retomada em tempo hábil;

15. **Conformidade:** adequar os requisitos de segurança para não permitir a violação de regulamentações ou leis estabelecidas, além de obrigações contratuais.

Entre estas 11 seções da Norma, temos uma que trata exclusivamente de segurança física para proteção da informação, que é justamente o tema central de nosso artigo. Esta seção possui 13 itens de controle para serem aplicados de acordo com a necessidade de cada organização. Elas estão distribuídas da seguinte forma:

9 Segurança física e do ambiente

9.1 Áreas seguras

9.1.1 Perímetro de segurança física.

9.1.2 Controles de entrada física.

9.1.3 Segurança em escritórios salas e instalações.

9.1.4 Proteção contra ameaças externas e do meio ambiente.

9.1.5 O trabalho em áreas seguras.

9.1.6 Acesso do público, áreas de entrega e de carregamento.

9.2 Segurança de equipamentos

9.2.1 Instalação e proteção do equipamento.

9.2.2 Utilidades.

9.2.3 Segurança do cabeamento.

9.2.4 Manutenção dos equipamentos.

9.2.5 Segurança de equipamentos fora das dependências da organização.

9.2.6 Reutilização e alienação segura de equipamentos.

9.2.7 Remoção de propriedade.

Com base nestes controles, vamos apresentar algumas sugestões de como sua organização pode elevar o nível de segurança. A implementação dos controles da seção **9 Segurança física e do ambiente** traz mais do que uma proteção para a informação. Estes controles contribuem para a proteção dos ativos que representam valor para a organização, incluindo equipamentos e recursos humanos. Com essa visão já começamos a mudar o ponto de vista, antes voltado apenas para o departamento de TI, mas agora abrindo o horizonte para mostrar os benefícios a toda organização, incluindo o bem estar das pessoas.

Áreas seguras

Em todas as organizações existem áreas em que é permitida a circulação livre de pessoas e materiais. Por outro lado, há também áreas restritas, onde a circulação de pessoas e materiais deve ser controlada para preservar algo de valor para a organização ou para garantir a segurança das pessoas.

Para implementar uma ou mais áreas seguras na organização, é necessário antes identificar quais ativos deseja-se proteger e por

quais razões. Isso é importante para garantir a proteção adequada ao valor do ativo. Não faz sentido construir um muro alto de concreto com cerca elétrica para proteger um barraco simples de madeira. Como também não adianta nada comprar um palácio decorado com ouro e não querer protegê-lo com muros para não ofuscar sua beleza.

Com estas informações bem definidas, podemos começar a adotar os controles necessários discutidos na sequência.

Perímetro de segurança física

O perímetro de segurança física é um delimitador, uma barreira que tem como objetivo controlar, restringir ou mesmo impedir o acesso a determinado local ou determinado objeto. Em uma organização, encontramos (ou deveríamos encontrar) uma combinação de barreiras para limitar o acesso a determinados pontos. As barreiras podem ser físicas ou combinadas com recursos lógicos. Exemplos de barreiras puramente físicas: portarias, recepções, cercas, muros, catracas comuns. No caso de barreiras com recursos lógicos empregados, temos como exemplos portas ou catracas com fechaduras eletrônicas abertas por crachás, senhas ou leitor biométrico, muito usado atualmente. Alarmes e Circuitos Internos de TV (CFTV) também são considerados barreiras que auxiliam na proteção do ambiente.

Estes recursos em sua maioria já são utilizados nas organizações, mas muitas vezes falta uma administração correta. Inúmeras vezes consegui entrar sem a devida identificação em empresas que possuíam portarias e recepções bem equipadas, mas com pessoas mal treinadas. Bastava um pouco de conversa e o meu acesso era liberado. Em outros locais, bastava entrar com o carro e dar um "tchauzinho" de longe, pois a cancela ficava sempre aberta para a entrada de veículos.

Já presenciei alguns casos em que toda a estrutura de CFTV estava lá, funcionando, gravando cada movimento, mas quando foi solicitada a exibição de uma determinada cena, o arquivo já havia sido sobregravado em um curto espaço de tempo. Adiantou investir e não gerenciar? Por isso verifique a necessidade de sua organização, aplique os perímetros e gerencie.

Controles de entrada física

Além dos perímetros para criar barreiras que dificultem o acesso à organização e que protejam os ativos em geral, incluindo as pessoas, é necessário um cuidado maior com os locais identificados como áreas seguras. Nestes locais o acesso só pode ser realizado por meio de credenciais que possam validar a presença da pessoa no ambiente, como o uso de crachás magnéticos, cartões eletrônicos ou senhas para fechaduras de portas e catracas eletrônicas. Atualmente vem se destacando neste setor o uso de biometria.

Além destes recursos tecnológicos, a adoção de procedimentos por parte de todos os funcionários da organização é fundamental. Todos devem estar cientes de suas responsabilidades, como manter sempre visível sua identificação e exigir de seus visitantes o mesmo, sejam eles clientes ou fornecedores. Todos os visitantes devem ser identificados em uma recepção, onde seus dados serão registrados e armazenados para eventuais consultas. Os visitantes devem ser supervisionados ou acompanhados por alguém responsável na organização.

Segurança em escritórios, salas e instalações

Com a descentralização das informações, que antes ficavam nos mainframes dos antigos CPDs e hoje se encontram fragmentadas em toda a rede, ficou difícil estabelecer limites físicos para proteger os ativos, especialmente nos dias atuais, onde a mobilidade virou requisito obrigatório em qualquer tipo de negócio. Portanto, este tipo de controle só tem validade quando combinado com outros controles que previnem o acesso da informação fora das áreas seguras.

De qualquer forma, ainda há diversos ambientes de trabalho com equipamentos fixos (desktops, impressoras, copiadoras, faxes, fitas DAT, pen drives) e onde a informação a ser protegida não está apenas no formato digital, mas em formato impresso também. Pensando nisso, a organização deve proteger o acesso a estes ambientes para evitar furto, roubo de ativos ou acesso a informações importantes. O uso de armários com fechaduras, política de mesa limpa, para não deixar documentos expostos a todo o momento e bloqueios de tela para evitar o acesso não autorizado aos sistemas da empresa durante a ausência do usuário são alguns exemplos de recursos que podem ser utilizados.

Proteção contra ameaças externas e do meio ambiente

Toda a organização deve mapear quais as ameaças externas e do meio ambiente ela está exposta, de acordo com as características da região e do setor de mercado da qual ela atua. Uma empresa localizada próxima à passagem de rios, deve adotar proteções específicas contra enchentes. Uma empresa que trabalha com produtos altamente inflamáveis, deve ter sua central de processamento em local afastado das áreas com maiores riscos, ou ainda empresas próximas a centros de detenções ou penitenciarias, devem ter planos de continuidades bem definidos, pois na realidade do nosso país, é comum a ocorrência de rebeliões que acabam isolando as áreas próximas, inclusive a empresa e seus funcionários.

Além destas precauções, a organização deve manter seus sistemas elétricos e hidráulicos em manutenções periódicas.

O trabalho em áreas seguras

Áreas seguras não devem ser identificadas facilmente. Parece meio que contraditório, mas se você analisar, somente quem necessita exercer suas atividades neste local é que tem que saber que aquele ambiente fechado é uma área segura. Vejo em muitas empresas placas indicando claramente "SALA DE SERVIDORES" ou "SALA DE MONITORAMENTO". Ninguém precisa saber que ali estão todos os seus servidores ou todos os registros de filmagem. Se um fornecedor necessitar prestar algum tipo de suporte nestes locais, ele deverá ser acompanhado por alguém que sabe onde ficam os equipamentos.

Todo o trabalho em área segura deve ser monitorado, principalmente quando se tratar de visitantes (fornecedores ou terceiros).

Estes devem ter acesso somente às informações necessárias para execução de seus trabalhos. Nestas áreas deve-se evitar o uso de câmeras fotográficas ou de filmagem, a não ser com expressa autorização da área responsável.

Acesso do público, áreas de entrega e de carregamento

Estas áreas onde circulam um grande número de pessoas devem ser, sempre que possível, afastadas das áreas seguras da empresa, para evitar o acesso não autorizado de entregadores, por exemplo. Locais onde são realizados atendimentos públicos devem ser protegidos para evitar o roubo de ativos ou acesso aos sistemas sem autorização.

Segurança de equipamentos

Os equipamentos que armazenam e processam as informações devem ser protegidos contra ameaças físicas e do ambiente, para garantir a tríade básica da segurança da informação (confidencialidade, disponibilidade e integridade) e consequentemente garantir a continuidade dos negócios, evitando assim prejuízos para a organização.

Neste item é que encontramos alguns dos principais argumentos para investir em tecnologia para atender aos requisitos da Norma ISO 27002. Mas vale lembrar que a aquisição de tecnologia sem o apoio de processos bem desenhados e da capacitação de pessoas não proverá os resultados esperados.

Instalação e proteção do equipamento

Todos os equipamentos devem ser instalados em ambientes adequados, protegidos contra ameaças físicas e do meio ambiente. Deixar equipamentos expostos à poeira, altas temperaturas, vibrações ou interferências elétricas podem comprometer sua durabilidade, e dessa forma afetar a integridade e a disponibilidade das informações. Muitos departamentos de TI adotam ambientes com controle de temperatura, umidade e de alimentação elétrica, mas às vezes esquecem detalhes como excesso de vibração e acomodação correta dos equipamentos. Já visitei salas de servidores que conforme andávamos sobre o piso elevado, fazíamos os equipamentos balançarem, justamente por estarem acomodados diretamente no piso. Em ambientes industriais, deve-se evitar instalar os servidores em locais que possam vibrar devido o impacto de prensas ou durante a passagem de veículos pesados.

Outros cuidados necessários são a restrição de entrada de alimentos, bebidas e materiais químicos e inflamáveis dentro do ambiente onde estão alocados os equipamentos de informática.

Utilidades

Todo o sistema que provê alimentação elétrica, suprimento de água e climatização do ambiente é identificado com nome de Utilidade. Estes sistemas permitem que os equipamentos que armazenam e processam as informações estejam sempre disponíveis, evitando paradas abruptas que poderiam comprometer a integridade dos dados.

O uso de sistemas de suprimento ininterrupto de energia ou UPS (*Uninterruptible Power Supply*) como no-breaks, por exemplo, é algo até que comum na realidade das pequenas e médias empresas. Mas isso por si só acaba não trazendo benefícios reais para a organização, pois se não forem tomadas medidas preventivas, o no-break acaba se tornando apenas mais um acessório na sala de servidores.

O no-break tem como principais funções estabilizar qualquer variação na alimentação de energia elétrica e manter por um breve período de tempo o fornecimento de energia, o suficiente para que todos os equipamentos a ele conectados possam ser desligados corretamente. Em situações em que os equipamentos não possam sofrer qualquer tipo de interrupção, se faz necessário o uso de geradores de energia. Em ambos os casos, a manutenção preventiva e a realização de testes periódicos para checar sua autonomia e sua capacidade é um requisito fundamental.

Apenas com o perfeito funcionamento das UPS, é que as demais Utilidades como ar-condicionado e sistemas de calefação poderão assegurar o nível adequado de climatização do ambiente e evitar as consequências de uma interrupção de abastecimento de energia.

Segurança do cabeamento

Por mais trivial que possa parecer, manter este requisito em plena conformidade é uma das tarefas mais difíceis devido a nossa cultura enraizada de "Disponibilidade a qualquer custo!". Com a crescente expansão dos equipamentos tecnológicos, a necessidade de disponibilizar pontos de rede e de telefone é essencial, porém diante da pressão em concluir as solicitações da organização, o que vemos é uma série de extensões das conexões em cascata, mal projetadas, que resultam em baixa qualidade do sinal, afetando a performance de toda a rede.

Isso acontece em muitos casos porque o próprio departamento de TI assume a responsabilidade de "puxar" cabos de rede para aumentar a distribuição de pontos, pois hoje quem não tem um alicate para crimpar conectores RJ-45? Só que saber "puxar" cabos e crimpar os conectores não é a única preocupação que se deve ter. O cabeamento lógico (rede e telefone) deve ser separado dos cabos que conduzem energia elétrica, para evitar interferências magnéticas, e em todo o seu percurso, o cabeamento lógico deve ser protegido em conduítes e canaletas para evitar interceptações não autorizadas.

Manter os cabos devidamente identificados e ter em mãos um diagrama de toda sua distribuição são requisitos obrigatórios para qualquer departamento de TI bem administrado. Pela minha experiência, creio que a maioria das organizações não possui esta documentação. É óbvio que este levantamento realizado por uma empresa especializada tem um custo expressivo, mas é neste ponto que a argumentação do gestor de TI deve fazer toda a diferença. Qual o impacto que o negócio pode sofrer caso um cabo de comunicação seja rompido ou danificado durante a realização de alguma obra ou reforma na empresa? Qual será o custo para a empresa ao solicitar uma manutenção ou expansão

da rede de comunicação, quando o fornecedor se deparar com o trabalho extra que ele terá para executar o serviço, justamente por não ter qualquer informação previamente em mãos? Quanto tempo a equipe de TI perde durante a realização de suporte técnico e manutenção na rede, tentando localizar possíveis "gargalos" que deixam a rede lenta? Como identificar pontos de rede sem a documentação e as ferramentas corretas?

Se um gestor conseguir mensurar estes custos, ele terá melhores condições de negociar com a alta direção os investimentos necessários para adequar sua estrutura dentro de um padrão aceitável.

Manutenção dos equipamentos

Infelizmente na cultura de nosso país, quando falamos em manutenção, somos levados a pensar em consertos, algo reativo, quando na verdade o objetivo da manutenção é preventivo. Evitar que os equipamentos que suportam toda a operação da organização parem de funcionar é um dos principais objetivos deste controle.

Todas as manutenções realizadas devem ser antecipadamente programadas, dentro de intervalos periódicos, e todas as ações devem ser registradas, para manter um histórico de eventuais problemas, que auxiliarão na tomada de decisões futuras.

Segurança de equipamentos fora das dependências da organização

Hoje este é um dos pontos mais críticos da segurança da informação. Manter um nível de proteção dentro dos limites físicos das organizações já é uma tarefa difícil, ainda mais quando exposto às diversas ameaças do ambiente externo. Neste aspecto, somente a combinação de ferramentas, processos e pessoas trará algum efeito positivo.

Com toda a evolução do conceito de mobilidade sendo aplicado aos negócios, é natural que algumas medidas preventivas devam ser tomadas para evitar que toda as medidas de proteção adotadas internamente também tenham algum efeito externamente. O uso de dispositivos móveis como notebooks, smartphones, celulares, câmeras digitais, pen drives e tablets expõem a disponibilidade, a integridade, mas principalmente a confidencialidade das informações. O roubo ou a perda destes dispositivos pode trazer graves consequências e prejuízos imensuráveis à organização.

Para reduzir os riscos dessa exposição e os graves impactos caso estas ameaças se concretizem, as seguintes medidas devem ser tomadas:

- Criptografar as informações contidas em dispositivos móveis;
- Fazer o uso de senhas fortes, inclusive em dispositivos como celulares;
- Manter uma cópia atualizada dos dados armazenados nos dispositivos;
- Transportar os equipamentos de forma discreta e protegida contra impactos e exposições a variações de temperaturas e agentes químicos;
- Treinar os usuários sobre as melhores práticas em segurança da informação e conscientizar a equipe do uso aceitável destes dispositivos.

Reutilização e alienação segura de equipamentos

Este controle passa despercebido em grande parte das empresas. Quando tratamos de reutilização e alienação (ou descarte) de equipamentos tecnológicos, as principais preocupações das organizações são de cumprir regulamentações ambientais, para evitar descarte de componentes que possam contaminar o meio ambiente e também em atender questões de responsabilidade social, realizando doações a ONGs e institutos educacionais. Mas tão importante como estas questões, está o fato de que muitas vezes informações sensíveis da empresa acabam sendo esquecidas nas unidades de armazenamento dos equipamentos como computadores e copiadoras.

O simples ato de formatar o disco rígido de um computador não é garantia de que a informação não possa ser recuperada. O mercado de recuperação de discos tem evoluído de forma crescente, tanto para o bem quanto para o mal. Estas ferramentas são facilmente encontradas na Internet e para quem não tem conhecimento técnico no assunto, tem a sua disposição inúmeras empresas especializadas neste tipo de serviço. Mesmo discos com problemas físicos podem ainda ter parte de suas informações restauradas.

Para evitar este tipo incidente, todo o equipamento que não for mais utilizado e que contenham dados gravados, devem ter suas unidades de armazenamento destruídas fisicamente, observando os cuidados com a segurança do funcionário. Para equipamentos que possuam informações sensíveis e forem reutilizados em outros departamentos ou doados para alguma ONG ou instituição, devem ter seus dados apagados com softwares específicos, evitando apenas o recurso básico de formatação. Mesmo que seja para doação, deve-se avaliar a importância das informações gravadas e, se for pertinente, doar o equipamento sem a unidade de armazenamento.

Remoção de propriedade

A retirada de equipamentos de dentro das organizações, seja para realocação ou para manutenção externa, deve ser feita de forma controlada, registrando sua saída e se for o caso, sua entrada ao retornar para a organização. Em muitas empresas, este controle é rigoroso, com emissão de nota fiscal de saída e com autorização formal para liberação na portaria. Porém, em outras empresas, por se ter um contato constante com a assistência técnica, a informalidade ganha espaço. Essa "facilitação" para liberar a saída do equipamento pode fazer com que dados sensíveis acabem sendo entregues em mãos de terceiros, que muitas vezes não possuem qualquer precaução para preservar a integridade e a confidencialidade das informações.

Todo este processo deve ser registrado e acompanhado por um responsável para evitar a perda de ativos para a organização.

Conclusão

Como pudemos ver, a Norma ISO 27002 é muito abrangente, abordando vários aspectos no tratamento da informação. Para o departamento de TI, existem várias recomendações que devem ser avaliadas para que a área possa dar suporte ao negócio. Quan-

do o departamento de TI consegue mensurar a importância de suas ações, ele consegue obter com maior facilidade os recursos necessários para prover os serviços com qualidade a toda organização.

Como base inicial de qualquer ação de melhoria no departamento de TI em relação à segurança, a seção 9 da Norma dispõe de várias recomendações que são mais fáceis de serem mensuradas, por se tratar de melhorias que são visualmente percebidas pelas demais áreas. Em resumo, seus resultados são práticos, diferentemente de ações que envolvem mudança cultural, como políticas e treinamentos, que só são percebidas ao longo do tempo. Mas todas estas ações precisam acontecer para que a segurança possa realmente agregar valor ao negócio, lembrando que a segurança da informação é sustentada por ferramentas, processos e pessoas.



Roberto Henrique
roberto@abctec.com.br
Roberto Henrique é consultor da ABCTec, com 13 anos de experiência na área de TI (Suporte, Gestão, Consultoria), especializado em análise de vulnerabilidades e no tratamento de incidentes de segurança da informação. Formado em Análise e Desenvolvimento de Sistemas, possui as certificações Microsoft MCP/MCDST, F-Secure Certified Sales Professional - FCSP, D-Link DBC. Membro do Comitê Técnico ABNT/CB21:CE27 sobre Segurança da Informação e membro do Núcleo de TI do CIESP – São Bernardo do Campo.

Referência

ABNT – Associação Brasileira de Normas Técnicas. ABNT NBR ISO/IEC 27002 – Tecnologia da informação – Técnicas de segurança – Código de prática para a gestão de segurança da informação. ABNT, 2005.

FÓRUM
DEVMEDIA

O lugar perfeito para você ficar por dentro de tudo o que acontece nas tecnologias do mercado atual



No novo Fórum da DevMedia você vai encontrar canais específicos de Delphi, ASP.NET, Java, Banco de Dados e Engenharia de Software; além de ter contato com profissionais qualificados da área para troca de informações, sugestões e muito mais.

ACESSE AGORA
www.devmedia.com.br/forum

DevMedia

Cloud Computing com o Windows Azure

Giuliano Moreno, Renato José Groffe e Tiago Valim

Resumo DevMan

De que se trata o artigo:

Neste artigo veremos as principais características do padrão computacional conhecido como cloud computing. Em seguida, será apresentada a iniciativa disponibilizada pela Microsoft dentro deste segmento: a plataforma Windows Azure.

Para que serve:

Soluções de cloud computing possibilitam, através da Internet, o compartilhamento de recursos computacionais como infraestrutura de servidores ou armazenamento de dados, por exemplo. Tais serviços são cobrados por parte das empresas provedoras com base nos níveis de utilização dos mesmos. A plataforma Windows Azure da Microsoft se encaixa neste perfil.

Em que situação o tema é útil:

A adoção de uma solução de cloud computing pode ser de grande valia para as organizações. Dentre os motivos que podem levar ao emprego deste tipo de arquitetura, pode ser citada a busca por uma maior flexibilidade para as operações de TI, assim como a redução de custos com pessoal especializado, hardware e software.

É inegável a importância atual da Tecnologia da Informação para o mundo dos negócios. O ambiente empresarial exige que novas soluções de software sejam desenvolvidas frequentemente, assim como aplicações já existentes se adaptem a mudanças. Tudo isto busca garantir a continuidade de uma organização ou, ainda, possibilitar meios para o aproveitamento de oportunidades ainda inexploradas.

Uma gama de sistemas exige, contudo, vultosos e constantes investimentos em infraestrutura de hardware e suporte. Além disso, diversos requisitos como disponibilidade, segurança e escalabilidade, devem ser levados em consideração.

Atentos a isto, grandes fornecedores de soluções de tecnologia desenvolveram serviços computacionais que podem ser utilizados por outras companhias, mediante prévio acordo entre as partes. Este modelo de serviço recebe o nome de *cloud computing* ou "computação nas nuvens".

A plataforma Azure, objeto de estudo deste artigo, representa a solução da Microsoft para este segmento e procura, através de uma série de serviços, oferecer uma solução flexível e que seja, ao mesmo tempo, de fácil implementação.

Cloud computing: uma visão geral

O termo *cloud computing* ou "computação nas nuvens" é definido, em termos gerais, como um modelo em que recursos computacionais são compartilhados através da Internet por provedores de serviços especializados.

Tais recursos, por sua vez, são oferecidos por meio de uma abordagem em que se cobra pelo nível de uso dos mesmos, mediante acordo prévio estabelecido entre fornecedor e a empresa contratante.

Os diversos tipos de serviços existentes podem ser classificados nas seguintes categorias:

- *Storage-as-a-service*: envolve recursos de armazenamento em disco remotamente;
- *Application-as-a-service* ou *Software-as-a-service*: sistemas de software hospedados na "nuvem";
- *Database-as-a-service*: acesso remoto a bancos de dados;
- *Security-as-a-service*: dispositivos de segurança como gerenciamento de usuários, disponibilizados através da "nuvem";
- *Testing-as-a-service*: mecanismos para testes de aplicações;
- *Infrastructure-as-a-service*: toda a infraestrutura de *data center* disponibilizada ao contratante;
- *Platform-as-a-service*: conjunto de recursos envolvendo soluções de hospedagem de aplicações, banco de dados, armazenamento, testes, segurança e infraestrutura. A plataforma Azure enquadra-se nesta classificação.

Soluções On-premise x Off-premise

Serviços computacionais e aplicações corporativas, levando-se em conta o paradigma de "computação nas nuvens", podem ser classificados em:

- *On-premise*: implantados e gerenciados por uma organização com seus próprios recursos de infraestrutura de TI;



• *Off-premise*: aplicações e serviços na nuvem, ou seja, hospedados em *data centers* de um provedor, sendo este último responsável por oferecer toda a infraestrutura necessária para atender às necessidades da organização.

Vantagens e pontos de atenção

Dentre os benefícios normalmente almejados e que podem levar, por sua vez, uma empresa a investir em soluções de *cloud computing*, convém destacar:

• Redução de custos ao eliminar a necessidade contínua e dispendiosa de compra de novos equipamentos, gastos com instalações e energia, licenças de software e pessoal qualificado para manter as operações;

• Como a cobrança dos serviços se faz mediante o uso dos mesmos, épocas de baixa demanda pela utilização de recursos podem se traduzir, por sua vez, em economia orçamentária para a organização;

• Alta disponibilidade dos recursos, mediante a adoção de medidas de contingência pelo provedor de um serviço. Um exemplo disto seria a redundância de servidores em que a falha num dispositivo pode ser tolerada, tudo isso graças à existência de outros computadores para atender o mesmo tipo de requisição;

• Escalabilidade, que se traduz na capacidade da infraestrutura utilizada estar preparada para demandas crescentes, o que pode ser obtido através da facilidade de expansão dos recursos utilizados como novos servidores, mais espaço de armazenamento, *load balancing* (balanceamento de carga), etc.;

• Os utilizadores dos serviços não necessitam se preocupar com a plataforma de hardware envolvida no suporte às suas aplicações. Preocupações com detalhes físicos são ocultadas por quem adquire uma solução de *cloud computing*, processo este conhecido como virtualização;

• Velocidade na implementação de novos projetos de software, uma vez que uma empresa ao adotar este paradigma utilizará melhor o tempo de sua equipe de Tecnologia, focando nas funcionalidades que são vitais no negócio e diminuindo a preocupação com a administração da infraestrutura que estaria envolvida.

Sendo assim, ao contratar uma solução de *cloud computing*, uma organização deverá levar em conta aspectos como:

• A segurança dos serviços oferecidos;

• Questões legais relativas a processos de auditoria envolvendo *logging* de transações de dados;

• SLAs (*Service-Level Agreements*/Acordos de níveis de serviço) que definam claramente as responsabilidades, garantias e riscos inerentes às atividades do provedor contratado.

Plataforma Windows Azure

A plataforma Windows Azure é uma solução de *cloud computing* lançada oficialmente pela Microsoft em Outubro de 2008. O objetivo da mesma é prover uma estrutura escalável e baseada na Internet, facilitando para tanto o processo de criação e implantação de aplicações e serviços na "nuvem".

O ambiente de desenvolvimento Microsoft Visual Studio .Net possui amplo suporte para a concepção e disponibilização de aplicativos na "nuvem", com uma abordagem que em nada difere daquela utilizada com softwares *on-premise*. Tudo isto possibilita um maior foco na criação de soluções, diminuindo assim a preocupação com detalhes de infraestrutura.

Esta plataforma é composta por três serviços principais:

• **Windows Azure:** sistema operacional na "nuvem" que atua como ambiente para execução de aplicações, além de contar com recursos de armazenamento de dados;

• **AppFabric:** mecanismo de *middleware* da plataforma Azure, com funcionalidades de service bus e controle de acesso;

• **SQL Azure:** serviço de banco de dados relacional na "nuvem" e que estende, assim, as capacidades oferecidas por servidores SQL Server convencionais.

Estes três serviços podem tanto ser empregados separadamente no desenvolvimento de novas soluções, quanto combinados, conferindo assim flexibilidade na sua utilização. O acesso aos mesmos ocorre através da criação de uma conta no Windows Live Id, além da vinculação desta última a cada um dos serviços desejados, mediante acesso ao site dos mesmos.

É importante ressaltar ainda, a possibilidade de se escolher em qual *data center* aplicações ou dados serão hospedados, técnica esta conhecida como geolocalização. Tal necessidade pode surgir em virtude de razões legais que obriguem o armazenamento de dados numa determinada localidade, fatores políticos ou mesmo questões de performance (uma aplicação executando numa maior proximidade geográfica de dados dos quais depende, por exemplo).

Como é de praxe entre os fornecedores de soluções de *cloud computing*, o custo de utilização da plataforma Azure dependerá dos serviços que foram contratados por um cliente, assim como do nível de utilização dos mesmos. O portal do Windows Azure na Internet possui mais detalhes a respeito disso.

Sistema Operacional Windows Azure

O sistema operacional Windows Azure pode ser considerado como um ambiente virtual, funcionando como uma "camada na nuvem". O mesmo é executado em múltiplas máquinas configuradas com o Windows Server 2008 R2. Tais equipamentos contam, por sua vez, com a tecnologia Hyper-V de suporte à virtualização, situados em *data centers* da Microsoft.

É importante mencionar que o Windows Azure funciona como uma abstração do hardware envolvido no processo, ocultando detalhes físicos das aplicações e serviços hospedados na "nuvem". Além disso, encontra-se dividido em três componentes principais:

• **Compute:** atua como um servidor de aplicações;

• **Storage:** oferece meios de armazenamento de dados, bem como serviços de troca de mensagens através de filas;

• **Management:** responsável pelo gerenciamento do ambiente de *cloud computing*, através do controle de toda a infraestrutura a ser utilizada. Estas atividades são desempenhadas por um software chamado Fabric Controller.

A **Figura 1** representa a arquitetura do sistema operacional Windows Azure, incluindo os diversos serviços que o constituem.

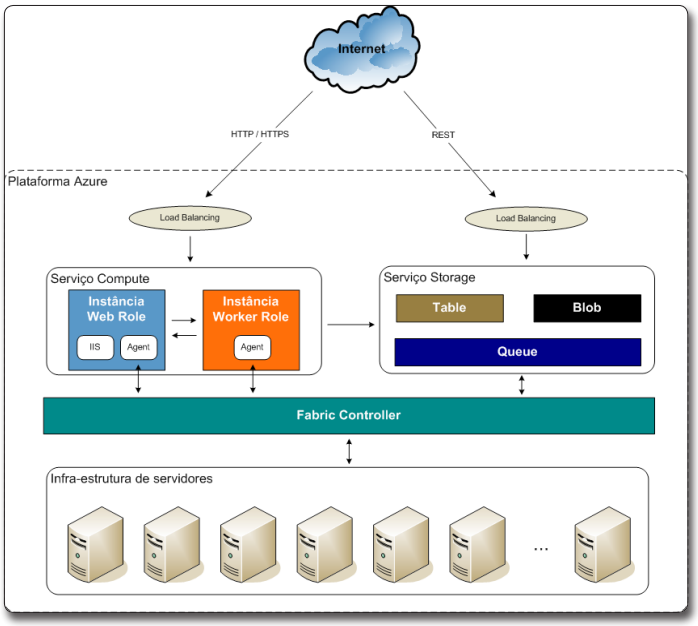


Figura 1. Arquitetura do Sistema Operacional Windows Azure

Componente Compute

O componente Compute permite a hospedagem e a execução de aplicações na "nuvem", utilizando-se da plataforma de 64 bits com a tecnologia Hyper-V, oferecida pelo Windows Server 2008.

Um dos objetivos principais da plataforma Azure é permitir um grande número de usuários simultâneos. Isso é possível através da criação de diversas instâncias para uma mesma aplicação, com cada uma destas executando sua própria VM (*virtual machine*). Cada VM, por sua vez, conta com um componente chamado Agent, que trabalha em estreita ligação com o Fabric Controller (este último é responsável pelo controle da infraestrutura e do ciclo de vida das instâncias criadas).

Este serviço permite a execução de dois tipos de aplicações (roles):

• **Web Roles:** softwares (sites, *web services*) que fazem uso do servidor de aplicações IIS (*Internet Information Services*) versão 7, recebendo normalmente requisições através dos protocolos HTTP ou HTTPS. Podem ser desenvolvidas utilizando-se tecnologias como ASP.NET ou WCF (*Windows Communication Foundation*), sob as versões 3.5 SP1 ou 4.0 do .Net Framework. É possível também a criação de *Web Roles* utilizando o módulo de extensão FastCGI, o qual possibilita a hospedagem de aplicações desenvolvidas em PHP ou C++, por exemplo;

• **Worker Roles:** correspondem a processos desenvolvidos para execução de forma contínua em background. Diferem das *Web Roles* por não serem hospedadas em servidores de aplicação IIS. Esta característica permite a utilização de serviços compatíveis com a plataforma Windows, mas que não foram desenvolvidos através da plataforma.Net, a exemplo de uma instalação de Apache Web Server em uma *Worker role*.

Web Roles, portanto, devem ser desenvolvidas como *stateless*, ou seja, sem manter informações de estado de um cliente. Tais informações, quando necessárias, serão persistidas no serviço de Storage da plataforma Azure ou, ainda, devolvidas instantaneamente a cada requisição. Isto se deve ao fato de ser impossível associar diversas requisições de um usuário a uma mesma instância de uma aplicação, já que o mecanismo de balanceamento de carga, empregado no tratamento das solicitações, não suporta este comportamento.

Desenvolvedores podem criar soluções que utilizem cada um dos tipos de roles definidos ou que combinem a utilização de ambos.

Componente Storage

O componente Storage da plataforma Azure possibilita o armazenamento de dados na "nuvem", bem como o acesso posterior aos mesmos, estando dividido em três serviços: Blob, Queue e Table.

Para se utilizar cada um dos tipos de Storage, deve-se fornecer uma conta de acesso e gerar requisições que atendam ao padrão REST por meio de uma API específica para cada serviço.

Em REST dados são representados por endereços únicos, através do uso de URLs. Um endereço único, por sua vez, é constituído por informações de identificação do dado no serviço de Storage, pela operação que se deseja efetuar sobre o mesmo e, ainda, pelo código da conta de acesso ao serviço.

O serviço Blob representa a forma mais simples de armazenamento, envolvendo a manipulação de dados binários (imagens, áudio, vídeos, etc.) sob a forma de arquivos. Blobs são agrupados logicamente em *containers*, podendo estes serem públicos (acessíveis para qualquer usuário) ou privados (visíveis somente ao proprietário da conta de acesso).

Existem dois tipos de Blobs:

• **Page Blob:** suporta até 1 TB. Uma simples operação de escrita neste tipo de blob resulta, por sua vez, no *commit* automático dos dados que foram transmitidos. É a opção mais indicada quando se busca rapidez na escrita e/ou leitura de imagens, vídeos, documentos, dentre outros;

• **Block Blob:** capacidade de armazenamento de até 200 GB. A escrita se faz mediante o envio de conjuntos menores de dados (*blocks*), cada um com no máximo 4 MB. A gravação definitiva (*commit*) em um blob pode ser feita após o upload de todos os *blocks* que constituem o mesmo. No caso de falhas, o reenvio de todos os *blocks* de um blob será evitado, continuando-se a partir do último trecho transmitido com sucesso.

A principal função do serviço Queue é fornecer meios para a comunicação assíncrona entre aplicações, através da troca de mensagens entre as mesmas. Embora o uso de *queues* (filas) sugira um comportamento restrito seguindo o padrão FIFO (*First In First Out*), não existe nenhuma garantia quanto à ordem de recebimento das mensagens dentro do sistema operacional Windows Azure. Além disso, uma mensagem não pode exceder o tamanho de 8 KB.

Queues representam um destino lógico para o envio de mensagens, ao passo que estas últimas, por sua vez, podem conter instruções e outras informações em formato texto ou mesmo binário.

Um uso possível para *queues* seria na comunicação entre instâncias *Web Role* e *Worker Role* de uma aplicação:
• Uma instância *Web Role*, em resposta a uma requisição, envia uma mensagem para uma *queue* específica;
• Uma instância *Worker Role* periodicamente acessa a *queue*, buscando mensagens a serem processadas. Ao obter uma dessas mensagens, as instruções necessárias são executadas, deletando-se posteriormente tal mensagem da *queue*.

O serviço Table possibilita o armazenamento de dados na "nuvem" sob a forma de tabelas. No entanto, o formato deste tipo de estrutura no Windows Azure difere bastante daquele existente em bancos de dados relacionais.

Uma tabela criada no ambiente de Storage é formada por um conjunto de entidades, com cada uma destas possuindo várias propriedades. Tais propriedades, por sua vez, podem ser do tipo **String**, **DateTime**, **Binary**, **Bool**, **GUID**, **Int** ou **Int64**.

Toda entidade conta, obrigatoriamente, com três propriedades:
• **PartitionKey:** identificação da partição em que uma entidade ficará armazenada;
• **RowKey:** identificador único de uma entidade dentro de uma partição;
• **Timestamp:** data / hora mantida pelo serviço Table.

O conteúdo das propriedades **PartitionKey** e **RowKey** pode ser definido livremente pelos desenvolvedores de uma aplicação.

A forma como uma tabela encontra-se estruturada no Azure permite à mesma armazenar terabytes de dados sob a forma de entidades. Já estas últimas, de acordo com o valor da **Partition Key**, podem estar particionadas em diversos servidores, como uma maneira de aumentar a performance no acesso a dados. Para acessar uma tabela as aplicações farão de forma transparente, sem a necessidade de conhecer em quais partições estão as entidades a serem manipuladas.

O acesso às tabelas de Storage do Azure não é realizada de maneira convencional (usando um driver/framework de acesso, além de instruções SQL), mas através de frameworks como ADO.NET Data Services ou LINQ (*Language Integrated Query*), os quais são capazes de se comunicar com o serviço Table via padrão REST.

Componente Management

O componente Management controla toda a infraestrutura básica, bem como oferece recursos de monitoramento das aplicações que se encontram na "nuvem". O hardware envolvido para a execução de uma aplicação ou serviço abrange servidores, switches, roteadores, equipamentos de balanceamento de carga, dentre outras estruturas. O controle de tais dispositivos ocorre através de um software denominado Fabric Controller.

É responsabilidade também deste componente o gerenciamento de todo o ciclo de vida de uma solução na "nuvem", permitindo aos serviços Compute e Storage atuar sem preocupações diretas com aspectos físicos.

Ademais, arquivos de configuração, específicos para cada aplicação implantada na rede, contêm parâmetros que orientam as ações do Fabric Controller (quantas instâncias de uma *role* deverão ser criadas e mantidas, por exemplo).

Constituem atribuições do Fabric Controller atividades como:
• Gerenciamento dinâmico e otimizado de hardware, incrementando ou reduzindo automaticamente a capacidade de recursos a serem empregados, mediante critérios configuráveis;
• Balanceamento de carga, com o redirecionamento automático das requisições às diversas instâncias previstas para uma role;
• Upgrade dinâmico de um serviço publicado na "nuvem", sem a necessidade de interrupção do mesmo;
• Disponibilização de uma nova instância de uma *role* quando detectadas falhas em outra que estiver em execução;
• Alocação automática de novos recursos para a execução de uma aplicação em resposta a falhas de hardware.

É possível ainda, dentro do serviço Management, a utilização de técnicas customizadas de *log*, *tracing* e monitoramento de nível do uso de aplicações.

AppFabric

O Windows Azure Platform AppFabric tem por função principal conectar aplicações e serviços, independentemente dos mesmos estarem na "nuvem" ou serem soluções *on-premise*. Todavia, não deve ser confundido com o componente Fabric do sistema operacional Windows Azure.

Diante destas características, é possível integrar aplicações nas mais diferentes plataformas, a exemplo do Windows Azure, Windows Server, Java, Ruby, PHP.

O componente AppFabric é constituído por dois serviços principais: Access Control e Service Bus.

Serviço Access Control

O serviço Access Control oferece mecanismos para o controle de acesso (autorização) em aplicações distribuídas, podendo estas últimas tanto representarem serviços na "nuvem", quanto soluções *on-premise*. Possibilita ainda a utilização de protocolos como REST, certificados X.509, recursos de Active Directory, dentre outros.

Existem dois métodos possíveis para se gerenciar o acesso:
• **Authorization roles:** papéis (*roles*), contendo cada um diferentes níveis de acesso. São definidos, posteriormente, os diversos usuários de um recurso a ser protegido, associados às roles das quais os mesmos farão parte, estabelecendo-se assim os direitos dos mesmos;
• **Claims based rules:** afirmações (*claims*). Definidas a fim de possibilitar o acesso a um determinado recurso. Este processo pode envolver variados tipos de dados como idade, domínio de e-mail, sexo, *roles* do usuário, etc., possibilitando, assim, um maior refinamento ao se estipularem as regras de autorização.

A **Figura 2** apresenta um exemplo de integração entre aplicações, utilizando o serviço Access Control, com um *token* sendo gerado para a aplicação cliente consumir o serviço de que necessita; esta última, por sua vez, compartilha com o Access Control uma chave com prazo de validade, que serve de base à validação do *token*.

Service Bus

O componente Service Bus representa, dentro da solução de *cloud computing* da Microsoft, a implementação do padrão de arquitetura conhecido como ESB (*Enterprise Service Bus*). Tem, por objetivo principal, possibilitar meios para uma conexão segura e de baixo acoplamento entre sistemas de software, permitindo aos mesmos superar barreiras que seriam impostas por firewalls ou outros dispositivos de hardware / software.

Permite ainda o uso de variados protocolos de comunicação no processo de integração entre aplicações, possibilitando a troca de informações entre softwares desenvolvidos nas mais variadas plataformas (Java, .Net, mainframe, etc.). HTTP, HTTPS, TCP, SOAP, MSMQ (*Microsoft Message Queuing*) e REST são alguns dos padrões disponíveis para utilização.

Serviços *on-premise* ou situados na "nuvem", podem ser registrados sob a forma de "*endpoints*" (endereços de acesso) no Service Bus. Uma aplicação que deseja consumir as funcionalidades oferecidas por um *endpoint* envia requisições ao mesmo, enquanto o Service Bus redireciona as chamadas aos endereços reais, devolvendo posteriormente os resultados a quem o invocou.

A compatibilidade entre o fornecedor e o consumidor de um conjunto de funcionalidades é assegurada, basicamente, através do uso de contratos (*interfaces*) detalhando as operações disponíveis; "*schemas*" descrevem a estrutura dos dados envolvidos na troca de mensagens entre aplicação-cliente e serviço.

A referida técnica oferece maior segurança, visto que o endereço real de um serviço registrado no Service Bus estará, forçosamente, oculto àqueles que venham a utilizar as operações do mesmo. O uso em conjunto com o componente Access Control do AppFabric tornará este processo ainda mais seguro.

Outro benefício desta metodologia consiste na facilidade em se descobrir serviços e capacidades destes, em virtude dos mesmos terem sido disponibilizados anteriormente através de *endpoints*.

A **Figura 3** apresenta, de maneira simplificada, o funcionamento do Service Bus dentro da plataforma Azure.

SQL Azure

O SQL Azure é um conjunto de serviços da plataforma de *cloud computing* da Microsoft, que procura, essencialmente, fornecer as capacidades de um sistema de gerenciamento de banco de dados relacional. Para tanto, o mesmo foi construído tomando por base o produto Microsoft SQL Server.

Dentre os principais benefícios deste componente da plataforma Azure, vale a pena destacar:

- Tarefas que envolvam a administração da infraestrutura (aquisição e configuração de hardware, configuração de servidores, etc.) não são mais necessárias por parte da organização que opta pelo SQL Azure. Isso ocorre porque o contrato firmado com a Microsoft confere a esta, a responsabilidade por tal função;
- Alta disponibilidade, uma vez que o mecanismo de redundância do SQL Azure gera e mantém atualizadas cópias de uma base de dados em diversos servidores. Com isso, se assegura a continuidade do negócio quando da ocorrência de graves falhas operacionais, além da economia do tempo gasto com atividades de *backup* e *restore*;
- Escalabilidade que se traduz na possibilidade de alocar mais recursos ou reduzir a infraestrutura utilizada, levando em conta a demanda do processo em questão;
- Facilidades na utilização dos recursos de banco de dados, uma vez que a manipulação de objetos da base se faz através da linguagem Transact-SQL (T-SQL), também existente no SQL Server;
- Ao utilizar o mesmo protocolo de acesso das versões *on-premise* do SQL Server, co-

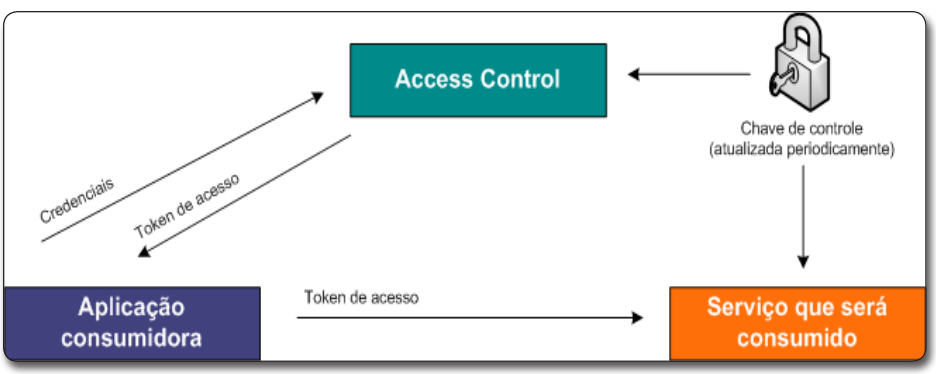


Figura 2. Serviço Access Control

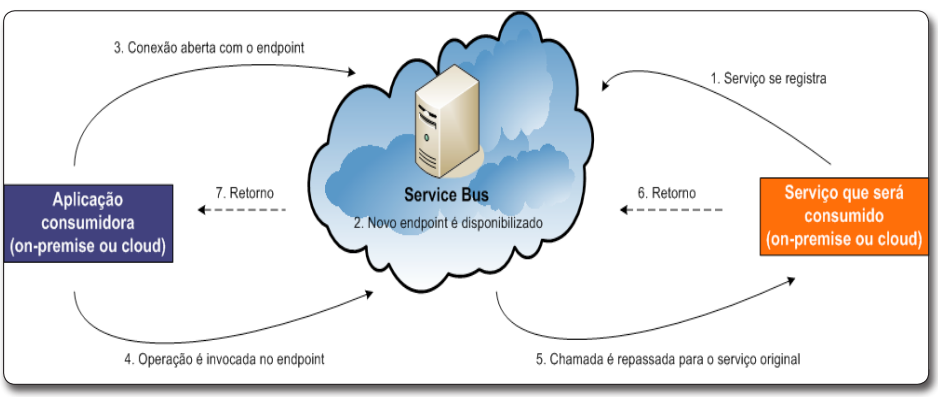


Figura 3. Service Bus

nhecido como TDS (*Tabular Data Stream*), aplicações (sejam *on-premise* ou *off-premise*) que suportem mecanismos de conexão como ADO.Net, .Net Entity Framework, ADO.Net Data Services, ODBC, JDBC, SQL for PHP e LINQ podem, de maneira transparente, acessar bases de dados situadas na "nuvem";

• Ferramentas como o SQL Server Management Studio e o utilitário de linha de comando SQLCMD.EXE também são permitidas, além de se contar com um portal para gerenciamento das bases de dados disponibilizado pela Microsoft.

O principal serviço dentro do SQL Azure chama-se SQL Azure Database (SAD). O SAD representa o meio através do qual ocorre a persistência de dados seguindo o modelo relacional.

No momento, os principais recursos de T-SQL como tabelas, índices, *views*, *stored procedures*, *triggers* e *constraints* podem ser utilizados sem restrições.

O diagrama representado na **Figura 4** apresenta dois cenários de utilização do SQL Azure: aplicações na nuvem (*Web* e *Worker roles*) acessando o banco de dados, assim como uma aplicação *on-premise* externa.

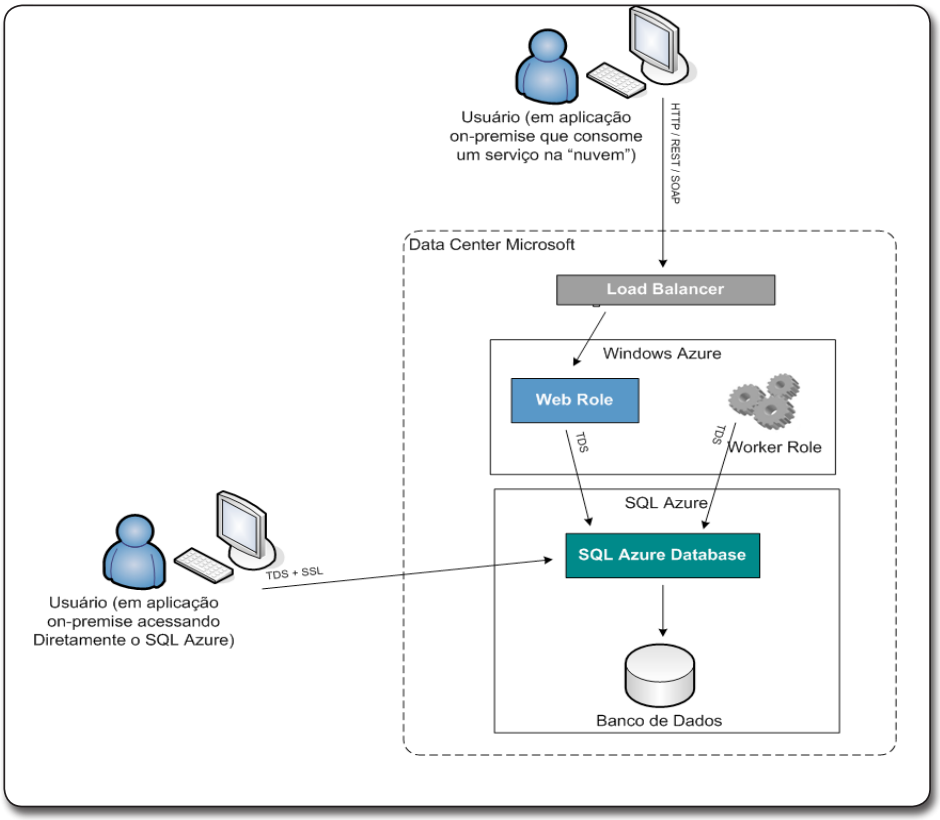


Figura 4. Aplicações acessando o SQL Azure

Outras soluções de cloud computing

Além do Windows Azure, merecem destaque como soluções de *cloud computing* as seguintes plataformas:

• **Amazon Web Services:** surgiram em 2006, quando a empresa de comércio eletrônico Amazon decidiu alocar sua infraestrutura de *data centers* a companhias interessadas na mesma. Dentre os diversos serviços oferecidos pela Amazon, há o ambiente virtual chamado Amazon EC2, que permite a utilização de diversos sistemas operacionais (Windows Server e diversas distribuições de Linux como Red Hat, Ubuntu, Fedora), servidores de bancos de dados e de aplicação (IIS/ASP.Net, Java Application Server, JBoss). Assim como o Windows Azure, possui um mecanismo próprio para banco de dados não-relacional (Amazon Simple DB);

• **Google App Engine:** é um ambiente para desenvolvimento e publicação de aplicações Web utilizando a infraestrutura da Google. Iniciou suas atividades em 2008 e oferece, atualmente, suporte às plataformas Java e Python, bem como disponibiliza *plugins* para a criação de soluções através da ferramenta Eclipse;

• **Force.com:** é uma plataforma para o desenvolvimento de aplicações corporativas distribuídas. Conta com uma IDE para desenvolvimento na Web, chamada Force.com Builder, além de um *plugin* para utilização por meio do Eclipse (Force.com IDE). Proporciona ainda funcionalidades para a realização de testes automatizados, assim como fórmulas e *workflows* pré-definidos voltados às áreas de CRM (*Customer Relationship Management*), SCM (*Supply Chain Management*) e ERP (*Enterprise Resource Planning*).

No Brasil, a LocaWeb, companhia de destaque no segmento de hospedagem de aplicações, também disponibiliza uma solução de *cloud computing*. Além dela, a operadora Telefônica, em parceria com a empresa de tecnologia NEC, anunciou desde o início de 2010, planos em que este tipo de serviço seria oferecido futuramente.

Análise de casos

Diversas empresas, nos mais variados segmentos de atuação, já fazem uso de soluções de *cloud computing* na plataforma Azure, a exemplo de 3M, Siemens e Kia Motors, grandes companhias de atuação global, que alcançaram sucesso.

Caso 1: 3M

Um dos projetos da 3M, denominado *Visual Attention Service* (VAS), consistia em uma aplicação Web que tornava possível a designers estimar, através de uma série de algoritmos de previsão, como suas criações poderiam ser vistas pelo tipo de audiência a que se destinavam.

Tais criações envolveriam desde a concepção de um logotipo, a um Web site ou, ainda, o local em que um painel eletrônico seria posicionado no *lobby* de um hotel. A partir de imagens, o serviço VAS pode sugerir (por meio de marcações) quais áreas do objeto em análise podem ser mais facilmente visualizadas e lembradas por um grupo de espectadores.

Buscando alta performance e disponibilidade, além de evitar vultosos investimentos em servidores e armazenamento, a 3M optou pela utilização da plataforma Azure. A aplicação resultante utiliza o .Net Framework, juntamente com os componentes, da plataforma Azure, a saber, o sistema operacional Windows Azure, Service Bus, Access Control Service e SQL Azure.

Caso 2: Siemens

Conhecido como cRSP (*common Remote Service Platform*), é um dos serviços desenvolvidos pela divisão de Tecnologia da Informação da Siemens. Trata-se de um sistema de serviços remotos, cujo gerenciamento utiliza uma grande infraestrutura disponibilizada por três *data centers* ao redor do globo.

A plataforma cRSP, utilizada pela Siemens, fornece suporte a mais de 80 mil dispositivos espalhados pelo mundo, permitindo que qualquer um destes equipamentos receba remotamente, via conexões VPN (*Virtual Private Network*), atualizações ou mesmo novas distribuições de software.

Em busca de mais agilidade, segurança, alta disponibilidade, redução de custos e complexidade, bem como de escalabilidade na utilização dos recursos, a companhia optou por utilizar a solução de *cloud computing* oferecida pela plataforma Azure. A nova solução de distribuição de software foi implementada empregando-se o .Net Framework 3.5, em conjunto com as tecnologias SQL Server, SQL Azure e Windows Azure. O armazenamento de pacotes de software a serem distribuídos ocorre por meio do uso conjunto do serviço Blob com o SQL Azure.

Caso 3: Kia Motors

Com o objetivo de promover o novo automóvel da companhia (o utilitário esportivo Kia Sorento 2011), a subsidiária norte-americana da Kia Motors decidiu pela veiculação de um comercial televisivo durante a final do campeonato de futebol americano (*Super Bowl*), em 7 de Fevereiro de 2010. No anúncio, possíveis consumidores eram estimulados a realizar um *test-drive* no veículo, devendo para isto efetuar a inscrição em um site promocional.

A campanha em questão tinha duração prevista para três meses. A organização pretendia evitar investimentos estimados em 100 mil dólares apenas com hardware, infraestrutura esta que entraria em desuso dado o caráter temporário da promoção.

Optou-se, assim, pelo desenvolvimento de uma solução utilizando o .Net Framework 3.5, com as tecnologias ASP.Net, SQL Azure e Windows Azure (sistema operacional). Inicialmente, o Web site criado contaria com 100 instâncias *Web Role*, com a possibilidade de se aumentar este número à medida que o número de acessos fosse aumentando. A referida estratégia permitiu que se atingisse o objetivo inicial: um recurso facilmente escalável e de baixo custo.

Desenvolvimento de aplicações com o Azure

O desenvolvimento de aplicações voltadas à plataforma Azure requer a instalação dos seguintes componentes:

- Windows Azure SDK (*framework* de desenvolvimento e runtime para execução de aplicações);
- Windows Azure Tools for Visual Studio (*plugin* para a criação de novos projetos).

O Windows Azure SDK encontra-se atualmente (Abril/2011) na versão 1.4, possuindo dois componentes para simulação de um ambiente de *cloud computing* em desenvolvimento:

- **Development Fabric:** emula o ambiente de execução a ser utilizado por *Web* e *Worker roles*;
- **Development Storage:** simula os serviços de Storage (Blob, Queue e Table).

Exemplo de utilização do Windows Azure

O desenvolvimento das aplicações apresentadas neste artigo foi realizado através da utilização do Microsoft Visual Studio 2010 Ultimate Edition, além da versão 1.2 dos pacotes Windows Azure SDK e Windows Azure Tools for Visual Studio.

O exemplo a seguir tem por objetivo demonstrar:

- Como se configuram aplicações para a utilização do Windows Azure;
- O desenvolvimento da plataforma Azure utilizando o Development Storage;
- A criação de uma *Web Role* usando um serviço em WCF;
- Como um serviço na nuvem pode ser consumido por uma aplicação externa.

Utilizando o Development Storage

O serviço de armazenamento local pode ser executado a partir do menu *Iniciar* do Windows, acessando-se para isto o submenu *Windows Azure SDK* e, em seguida, a opção *Development Storage*.

A **Figura 5** apresenta este serviço já em execução (nota-se que ao lado do status de cada serviço há o endereço/*endpoint* para o envio de requisições ao mesmo).

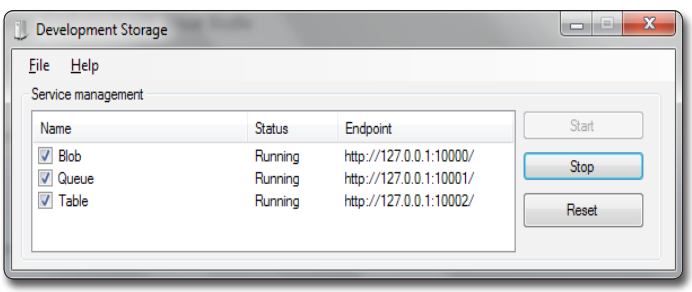


Figura 5. Development Storage

Para testar o serviço Table do Development Storage, se faz necessário criar uma solução no Visual Studio chamada *GeracaoDadosAzure*. O objetivo da mesma é a gravação de dados fictícios sobre ações negociadas em Bolsas de Valores. Esta solução será composta, por sua vez, pelos seguintes projetos:

- **StorageClient:** conjunto de classes para utilização do Storage do Windows Azure. É disponibilizado para download pela Microsoft no site da plataforma;

• **Exemplo.Cloud.Entities:** fará parte do mesmo a classe que representa a entidade a ser persistida através do serviço Table;
• **Exemplo.Cloud.DataAccess:** contém a classe para acesso à tabela de ações a ser criada no Development Storage;
• **GeracaoDadosAzure:** aplicação Web *on-premise* para carga de dados no serviço Table do Development Storage.

Na **Figura 6** é apresentada a estrutura da solução *GeracaoDadosAzure* dentro do Visual Studio, assim que o desenvolvimento da mesma tenha sido concluído.

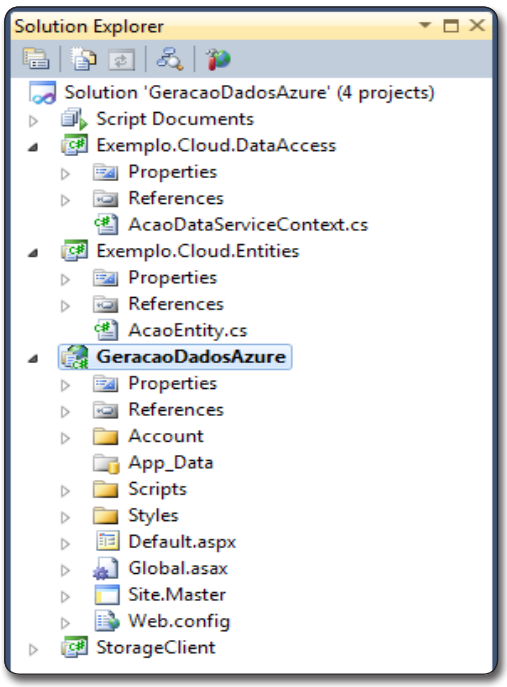


Figura 6. Estrututura da solução GeracaoDadosAzure

O projeto **StorageClient** foi baixado através do site da Microsoft, tendo sido adicionada uma referência ao mesmo na solução aqui apresentada.

Exemplo.Cloud.Entities é um projeto do tipo Class Library. No mesmo foi criada a classe **AcaoEntity** a qual define, por sua vez, a estrutura da entidade ação utilizada neste exemplo.

AcaoEntity (que está detalhada na **Listagem 1**) implementa a classe abstrata **TableStorageEntity**, sendo esta última a representação de uma entidade (linha) numa tabela. **TableStorageEntity** está definida dentro do projeto **StorageClient**, contando com as propriedades **PartitionKey**, **RowKey** e **Timestamp**. Convém ressaltar que estas três propriedades são obrigatórias para toda estrutura criada no serviço Table.

O projeto **Exemplo.Cloud.DataAccess**, também uma Class Library, contém a classe **AcaoDataServiceContext**. A mesma é responsável por fornecer os métodos que possibilitam a manipulação de entidades correspondentes a ações.

TableStorageDataService é uma classe abstrata implementada por **AcaoDataServiceContext**, estando definida no projeto **StorageClient**. Esta classe disponibiliza, por meio de uma série de operações, mecanismos de acesso aos dados contidos no serviço Table do componente de Storage.

Listagem 1. Código da classe AcaoEntity.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Samples.ServiceHosting.StorageClient;

namespace Exemplo.Cloud.Entities
{
  public class AcaoEntity : TableStorageEntity
  {
    public AcaoEntity()
      : base()
    {
      this.Timestamp = DateTime.Now;
    }

    public DateTime DataCotacao { get; set; }
    public String NomeEmpresa { get; set; }
    public String Moeda { get; set; }
    public Double ValorAtualAcao { get; set; }
    public Double ValorAnterior { get; set; }
    public Double PercentualVariacao { get; set; }
  }
}
```

AcaoDataServiceContext, definida na **Listagem 2**, faz uso do tipo **AcaoEntity**, que está contido no projeto **Exemplo.Cloud.Entities**. Utiliza ainda as seguintes classes definidas em StorageClient:
• **StorageAccountInfo:** representação de uma conta de acesso ao Windows Azure;
• **TableStorage:** fornece operações para a interação com o serviço Table.

A interface gráfica para a carga dos dados está contida no projeto **GeracaoDadosAzure**. Este último foi criado a partir do template "Web Application", dentro do Visual Studio, e faz referência a outros projetos já mencionados anteriormente, a saber, **StorageClient**, **Exemplo.Cloud.Entities** e **Exemplo.Cloud.DataAccess**.

O acesso ao Development Storage deve ser configurado no *Web.config* da aplicação **GeracaoDadosAzure**, que é automaticamente gerado ao se criar uma nova "Web Application". Os parâmetros necessários para a utilização do Windows Azure estão demonstrados na **Listagem 3** (outras configurações geradas de maneira automática foram omitidas, para efeitos de simplificação).

Os valores dos itens especificados na **Listagem 3**, referem-se aos endereços e demais configurações para simulação através do Development Storage. Caso esta aplicação acessasse diretamente os serviços do Azure, seriam configurados na mesma os endereços disponibilizados pela Microsoft, bem como a conta do Windows Live Id para uso do ambiente de cloud computing.

Tem-se na **Listagem 4** o código da página *Default.aspx*, no qual foram definidas as funcionalidades para criação de uma tabela no Windows Azure, carga inicial e visualização dos dados carregados.

Listagem 2. Código da classe AcaoDataServiceContext.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data.Services.Client;
using Microsoft.Samples.ServiceHosting.StorageClient;
using Exemplo.Cloud.Entities;

namespace Exemplo.Cloud.DataAccess
{
  public class AcaoDataServiceContext :
    TableStorageDataServiceContext
  {
    private const string TableName = "AcaoTable";
    private StorageAccountInfo accountInfo;

    public AcaoDataServiceContext(
      StorageAccountInfo accountInfo)
      : base(accountInfo)
    {
      this.accountInfo = accountInfo;
    }

    public void CriarTabela()
    {
      TableStorage ts =
        TableStorage.Create(this.accountInfo);
      if (!ts.DoesTableExist(
         AcaoDataServiceContext.TableName))
         ts.CreateTable(
         AcaoDataServiceContext.TableName);
      else
        throw new Exception(
          "Tabela de ações já criada");
    }

    public void IncluirAcao(AcaoEntity acao)
    {
      this.AddObject(
        AcaoDataServiceContext.TableName, acao);
      this.SaveChanges();
    }

    public List<AcaoEntity> ListarAcoesPorBolsa(
      string codigoBolsaValores)
    {
      var resultado =
        from acao in
        this.CreateQuery<AcaoEntity>(TableName)
        where acao.PartitionKey ==
            codigoBolsaValores
        select acao;
      return resultado.ToList();
    }

    public List<AcaoEntity> ListarTodasAcoes()
    {
      var resultado =
        from acao in
        this.CreateQuery<AcaoEntity>(TableName)
        select acao;
      return resultado.ToList();
    }
  }
}
```

Listagem 3. Web.config – Configurações de acesso ao Azure.

```
<configuration>

 ...

 <appSettings>

  ...

  <add key="AccountName"
     value="devstoreaccount1"/>
  <add key="AccountSharedKey"
     value="Eby8vdM02xNOcqFlqUwJPLlmEtlCDXJ1OUzFT50uSRZ
     6IFsuFq2UVErCz4I6tq/K1SZFPTOtr/KBHBeksoGMGw=="/>
  <add key="BlobStorageEndpoint"
     value="http://127.0.0.1:10000"/>
  <add key="QueueStorageEndpoint"
     value="http://127.0.0.1:10001"/>
  <add key="TableStorageEndpoint"
     value="http://127.0.0.1:10002"/>
  <add key="ContainerName"
     value="devstoreaccount1"/>

  ...

 </appSettings>

 ...

</configuration>
```

Conforme apresentado na **Listagem 4**, a propriedade **Partition Key** de cada entidade, que representa uma ação, foi preenchida com o código da Bolsa de Valores (LOND – Londres, NY – Nova York, SP – São Paulo) à qual a mesma se refere. Já no caso da propriedade **RowKey**, utilizou-se como conteúdo o código da ação.

Conforme a informação dada anteriormente, a performance no acesso a entidades pode ser aumentada através do uso de diferentes PartitionKeys.

Na **Figura 7** é apresentada a aplicação **GeracaoDadosAzure** já em execução, através do acesso à sua interface gráfica via browser. No caso, as opções *Criar Tabela* e *Preencher Tabela* foram previamente executadas. Após isto, clicou-se no botão *Consultar Dados*, tendo esta ação retornado todos os registros que constam na tabela **AcaoTable**, gerada no serviço de Storage local.

O ambiente de desenvolvimento Microsoft Visual Studio .Net possui amplo suporte para a concepção e disponibilização de aplicativos na "nuvem", com uma abordagem que em nada difere daquela utilizada com softwares on-premise. Possibilitando, portanto, um maior foco na criação de novas soluções, diminuindo assim a preocupação com detalhes de infraestrutura.

Criação de uma Web Role

Para criar e testar uma Web Role, utilizando o Development Fabric, deve-se, primeiramente, executar o Visual Studio com privilégios de Administrador. Isto pode ser feito no Windows 7, clicando-se com o botão direito do mouse sobre o ícone do Visual Studio, no menu Iniciar; em seguida, seleciona-se a opção *Executar como administrador*.

Listagem 4. Implementação da página Default.aspx

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Drawing;
using Microsoft.Samples.ServiceHosting.StorageClient;
using Exemplo.Cloud.DataAccess;
using Exemplo.Cloud.Entities;

namespace GeracaoDadosAzure
{
  public partial class Default : System.Web.UI.Page
  {
    protected void btnCriarTabela_Click(
      object sender, EventArgs e)
    {
      try
      {
        CriarTabelaAcoes();
        ConfigurarMensagem(
          "Tabela de ações criada com sucesso",
          false);
      }
      catch (Exception ex)
      {
        ConfigurarMensagem(
          "Erro durante a criação da " +
          "tabela de ações: " +
           ex.Message, true);
      }
    }

    protected void btnPreencherTabela_Click(
      object sender, EventArgs e)
    {
      try
      {
        PreencherTabelaAcoes();
        ConfigurarMensagem(
          "Tabela de ações preenchida " +
          "com sucesso", false);
      }
      catch (Exception ex)
      {
        ConfigurarMensagem(
          "Erro durante o preenchimento da " +
          "tabela de ações: " +
           ex.Message, true);
      }
    }

    protected void btnConsultarDados_Click(
      object sender, EventArgs e)
    {
      try
      {
        ConsultarTabelaAcoes();
        ConfigurarMensagem(
          "Consulta à tabela de ações " +
          "efetuada com sucesso", false);
      }
      catch (Exception ex)
      {
        ConfigurarMensagem(
          "Erro durante a consulta à tabela " +
          "de ações: " + ex.Message, true);
      }
    }

    private AcaoDataServiceContext CreateContext()
    {
     StorageAccountInfo accountInfo =
      StorageAccountInfo.
      GetDefaultTableStorageAccountFromConfiguration();

     return new AcaoDataServiceContext(accountInfo);
    }

    private void CriarTabelaAcoes()
    {
      AcaoDataServiceContext contextAcoes =
        this.CreateContext();
      contextAcoes.CriarTabela();
    }

    private void PreencherTabelaAcoes()
    {
      AcaoDataServiceContext contextAcoes =
        this.CreateContext();
      DateTime dataCotacao = new DateTime(
        2010, 8, 10);
      AcaoEntity acao;

      // Criação de dados para a Bolsa de Londres
      acao = new AcaoEntity()
      {
        PartitionKey = "LOND",
        RowKey = "STEEO",
        DataCotacao = dataCotacao,
        NomeEmpresa = "Steel Factories Inc.",
        Moeda = "Euro",
        ValorAtualAcao = 44.05,
        ValorAnterior = 43.8,
        PercentualVariacao = 0.57
      };
      contextAcoes.IncluirAcao(acao);

      acao = new AcaoEntity()
      {
        PartitionKey = "LOND",
        RowKey = "WEBL1",
        DataCotacao = dataCotacao,
        NomeEmpresa = "Web Learning Corporation",
        Moeda = "Euro",
        ValorAtualAcao = 44.55,
        ValorAnterior = 42.44,
        PercentualVariacao = 4.97
      };
      contextAcoes.IncluirAcao(acao);


      // Criação de dados para a Bolsa de Nova York
      acao = new AcaoEntity()
      {
        PartitionKey = "NY",
        RowKey = "BLUS1",
        DataCotacao = dataCotacao,
        NomeEmpresa = "Blue Sky Corporation",
        Moeda = "US$",
        ValorAtualAcao = 62.4,
        ValorAnterior = 60.47,
        PercentualVariacao = 3.19
      };
      contextAcoes.IncluirAcao(acao);

      acao = new AcaoEntity()
```

Continuação: Listagem 4. Implementação da página Default.aspx

```
        {
            PartitionKey = "NY",
            RowKey = "FAST2",
            DataCotacao = dataCotacao,
            NomeEmpresa = "Fast Aerolines Corporation",
            Moeda = "US$",
            ValorAtualAcao = 52.11,
            ValorAnterior = 52.97,
            PercentualVariacao = -1.62
        };
        contextAcoes.IncluirAcao(acao);

        acao = new AcaoEntity()
        {
            PartitionKey = "NY",
            RowKey = "SPEE0",
            DataCotacao = dataCotacao,
            NomeEmpresa = "Speed Stores Inc.",
            Moeda = "US$",
            ValorAtualAcao = 56.03,
            ValorAnterior = 55.89,
            PercentualVariacao = 0.25
        };
        contextAcoes.IncluirAcao(acao);


        // Criação de dados para a Bolsa de São Paulo
        acao = new AcaoEntity()
        {
            PartitionKey = "SP",
            RowKey = "ACME1",
            DataCotacao = dataCotacao,
            NomeEmpresa = "Empresas Acme S. A.",
            Moeda = "R$",
            ValorAtualAcao = 33.87,
            ValorAnterior = 32.78,
            PercentualVariacao = 3.33
        };
        contextAcoes.IncluirAcao(acao);

        acao = new AcaoEntity()
        {
            PartitionKey = "SP",
            RowKey = "AUTO1",
            DataCotacao = dataCotacao,
            NomeEmpresa =
                "Empresa Brasileira de Automóveis S. A.",
            Moeda = "R$",
            ValorAtualAcao = 25.83,
            ValorAnterior = 25.21,
            PercentualVariacao = -2.4
        };
        contextAcoes.IncluirAcao(acao);

        acao = new AcaoEntity()
        {
            PartitionKey = "SP",
            RowKey = "BORR1",
            DataCotacao = dataCotacao,
            NomeEmpresa =
                "Empresa Brasileira de Borrachas S. A.",
            Moeda = "R$",
            ValorAtualAcao = 20.71,
            ValorAnterior = 21.28,
            PercentualVariacao = 2.75
        };
        contextAcoes.IncluirAcao(acao);
    }

    private void ConsultarTabelaAcoes()
    {
        AcaoDataServiceContext contextAcoes =
            this.CreateContext();
        gridAcoes.DataSource =
            contextAcoes.ListarTodasAcoes();
        gridAcoes.DataBind();
    }

    private void ConfigurarMensagem(
        string mensagem, bool erro)
    {
        lblMensagem.Visible = true;
        lblMensagem.Text = mensagem;
        if (!erro)
            lblMensagem.ForeColor = Color.Black;
        else
            lblMensagem.ForeColor = Color.Red;
    }
  }
}
```

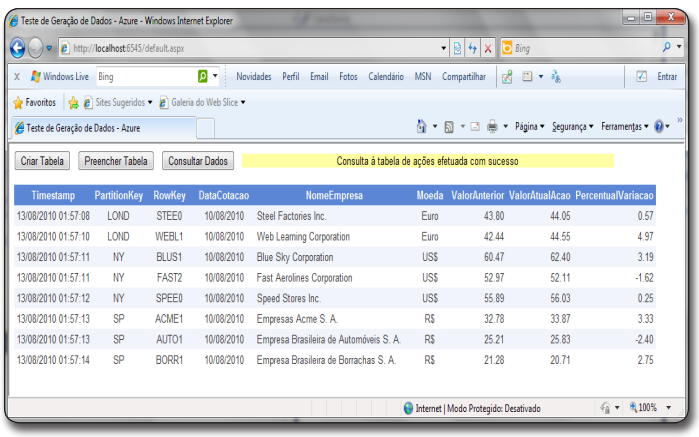


Figura 7. Dados já carregados através da aplicação GeracaoDadosAzure

A solução a ser criada chama-se *ServicoAzure* e fará uso do template "Cloud" (instalado através do pacote Windows Azure Tools for Visual Studio), conforme indicado na **Figura 8**.

Como próximo passo, é necessário selecionar o template de projeto a ser empregado para o desenvolvimento da Web Role. No caso, será criado um serviço chamado *WCFServiceAcoes*, tendo sido escolhida a opção *WCF Service Web Role* (**Figura 9**).

WCFServiceAcoes é um Web Service que faz uso da tecnologia WCF (*Windows Communication Foundation*), sendo esta última uma solução da Microsoft para a criação de aplicações distribuídas. O referido serviço visa, basicamente, simular uma aplicação com três instâncias em execução, responsável por retornar cotações de ações comercializadas em uma Bolsa de Valores.

Na **Figura 10** é apresentada a estrutura da solução *ServicoAzure* dentro do Visual Studio, tão logo o desenvolvimento da mesma tenha sido finalizado.

O projeto *ServicoAzure* é gerado durante a criação da solução, possuindo uma referência para a role que faz parte da mesma (no caso, *WCFServiceAcoes*). Além disso, contém os seguintes arquivos de configuração:

- *ServiceDefinition.csdef*: define um conjunto de propriedades configuráveis para a solução, tais como conta de acesso, endereços do serviço de Storage, etc. (**Listagem 5**);
- *ServiceConfiguration.csdef*: contém os valores especificados para cada propriedade configurável da solução (**Listagem 6**).

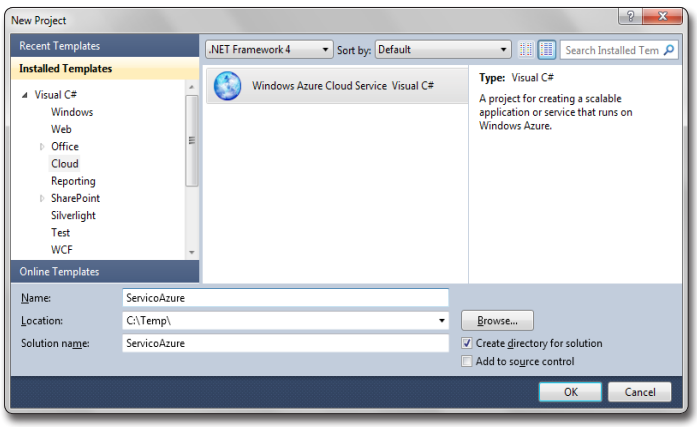


Figura 8. Criação da solução ServicoAzure

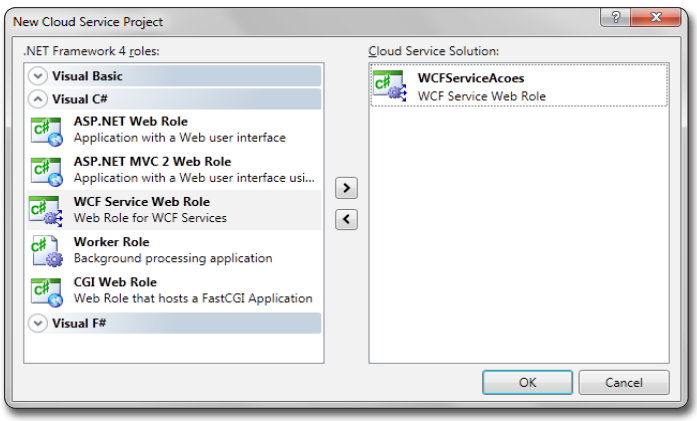


Figura 9. Criação do projeto WCFServiceAcoes

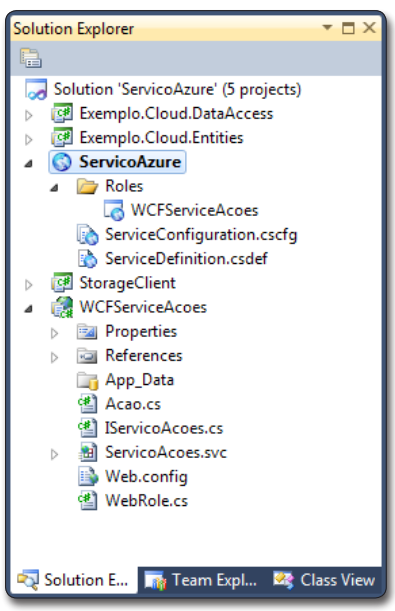


Figura 10. Estrutura da solução ServicoAzure

Listagem 5. Arquivo ServiceDefinition.csdef.

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceDefinition name="ServicoAzure"
  xmlns="http://schemas.microsoft.com/ServiceHosting/
    2008/10/ServiceDefinition">
 <WebRole name="WCFServiceAcoes">
  <InputEndpoints>
   <InputEndpoint name="HttpIn" protocol="http"
     port="80" />
  </InputEndpoints>
  <ConfigurationSettings>
   <Setting name="DiagnosticsConnectionString" />
   <Setting name="AccountName"/>
   <Setting name="AccountSharedKey"/>
   <Setting name="BlobStorageEndpoint"/>
   <Setting name="QueueStorageEndpoint"/>
   <Setting name="TableStorageEndpoint"/>
   <Setting name="ContainerName"/>
  </ConfigurationSettings>
 </WebRole>
</ServiceDefinition>
```

Listagem 6. Arquivo ServiceConfiguration.csdef.

```
<?xml version="1.0"?>
<ServiceConfiguration serviceName="ServicoAzure"
  xmlns="http://schemas.microsoft.com/ServiceHosting/
    2008/10/ServiceConfiguration">
 <Role name="WCFServiceAcoes">
  <Instances count="3" />
  <ConfigurationSettings>
   <Setting name="DiagnosticsConnectionString"
     value="UseDevelopmentStorage=true" />
   <Setting name="AccountName"
     value="devstoreaccount1" />
   <Setting name="AccountSharedKey"
     value="Eby8vdM02xNOcqFlqUwJPLlmEtlCDXJ1
       OUzFT50uSRZ6IFsuFq2UVErCz4I6tq/K1SZ
       FPTOtr/KBHBeksoGMGw==" />
   <Setting name="BlobStorageEndpoint"
     value="http://127.0.0.1:10000" />
   <Setting name="QueueStorageEndpoint"
     value="http://127.0.0.1:10001" />
   <Setting name="TableStorageEndpoint"
     value="http://127.0.0.1:10002/" />
   <Setting name="ContainerName"
     value="devstoreaccount1" />
  </ConfigurationSettings>
 </Role>
</ServiceConfiguration>
```

Configurar o número de instâncias da Web Role *WCFServiceAcoes* a serem executadas, é um procedimento muito simples, conforme indicado na **Figura 11**, podendo ser feito por meio de um duplo clique, sobre a referência para a mesma no projeto *ServicoAzure*.

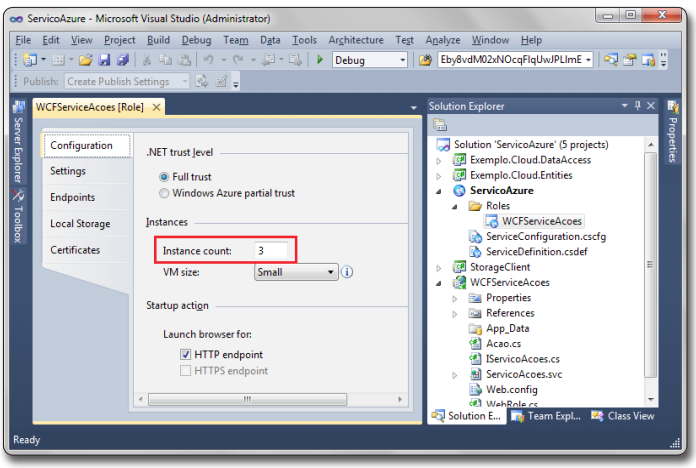


Figura 11. Configurando o número de instâncias da role WCFServiceAcoes

Deste modo, definiu-se (**Figura 11**) que serão criadas três instâncias do serviço *WCFServiceAcoes* quando da execução do mesmo, via Development Fabric.

O projeto *WCFServiceAcoes* representa a implementação em WCF da Web Role a ser hospedada na "nuvem". Constituído por uma interface chamada **IServicoAcoes**, bem como pelas classes **Acao** e **ServicoAcoes**, faz uso das bibliotecas **StorageClient**, **Exemplo.Cloud.Entities** e **Exemplo.Cloud.DataAcess**.

A classe **Acao**, apresentada na **Listagem 7**, é um tipo de dado complexo, conhecido em WCF como "Data Contract", representando uma cotação negociada em Bolsa de Valores.

A classe **ServicoAcoes**, implementação da interface **IServicoAcoes** e apresentada na **Listagem 9**, servirá de base para a geração das diversas instâncias da Web Role *WCFServiceAcoes*. Será responsável, ainda, pelo processamento de requisições enviadas por consumidores do serviço.

Já **IServicoAcoes** (**Listagem 8**), representa um tipo de interface conhecido em WCF como "Service Contract". O referido tipo serve de base, dentro desta tecnologia, à geração do arquivo WSDL contendo as especificações de um serviço.

Com todas as configurações, classes e interfaces já definidas, chega o momento de se executar o Web Service para consulta de ações. O mesmo deve ser ativado, para efeitos de simulação, através do Visual Studio. Para isto deve-se acessar a opção *Start Debugging* do menu *Debug*.

A **Figura 12** apresenta o serviço Development Fabric com a Web Role *WCFServiceAcoes* já em execução. Nota-se que foram criadas três instâncias do Web Service, numeradas de 0 a 2.

O endereço *http://127.0.0.1:81/ServicoAcoes.svc* será disponibilizado pelo Development Fabric automaticamente, a fim de possibilitar testes com o serviço *WCFServiceAcoes*. Acessando este Web Service, através do Internet Explorer, será exibida uma página com instruções para o consumo do mesmo (**Figura 13**).

Listagem 7. Código da classe Acao.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Runtime.Serialization;

namespace WCFServiceAcoes
{
  [DataContract]
  public class Acao
  {
    [DataMember]
    public String CodigoBolsaValores { get; set; }

    [DataMember]
    public String CodigoAcao { get; set; }

    [DataMember]
    public DateTime DataCotacao { get; set; }

    [DataMember]
    public String NomeEmpresa { get; set; }

    [DataMember]
    public String Moeda { get; set; }

    [DataMember]
    public Double ValorAtualAcao { get; set; }

    [DataMember]
    public Double ValorAnterior { get; set; }

    [DataMember]
    public Double PercentualVariacao { get; set; }
  }
}
```

Listagem 8. Código da interface IServicoAcoes.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;

namespace WCFServiceAcoes
{
  [ServiceContract]
  public interface IServicoAcoes
  {
    [OperationContract]
    List<Acao> ListarAcoesPorBolsa(
      string codigoBolsaValores);
  }
}
```

Consumindo um serviço na "nuvem"

Com a *Web Role WCFServiceAcoes* em execução, será necessária a abertura de uma nova instância do Visual Studio, a fim de se consumir a mesma em uma aplicação de teste chamada *ConsumidorServicoAzure*.

Listagem 9. Código da classe ServicoAcoes.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;
using Microsoft.Samples.ServiceHosting.StorageClient;
using Exemplo.Cloud.DataAccess;
using Exemplo.Cloud.Entities;

namespace WCFServiceAcoes
{
  public class ServicoAcoes : IServicoAcoes
  {
    public List<Acao> ListarAcoesPorBolsa(
      string codigoBolsaValores)
    {
      List<Acao> listResultado = new List<Acao>();
      Acao acao;

      StorageAccountInfo accountInfo =
        StorageAccountInfo.
        GetDefaultTableStorageAccountFromConfiguration();
      AcaoDataServiceContext context =
        new AcaoDataServiceContext(accountInfo);
      List<AcaoEntity> listCloudAcoes =
        context.ListarAcoesPorBolsa(
          codigoBolsaValores);

      foreach (AcaoEntity entidadeCloud
          in listCloudAcoes)
      {
        acao = new Acao();
        acao.CodigoBolsaValores =
          entidadeCloud.PartitionKey;
        acao.CodigoAcao =
          entidadeCloud.RowKey;
        acao.DataCotacao =
          entidadeCloud.DataCotacao;
        acao.NomeEmpresa =
          entidadeCloud.NomeEmpresa;
        acao.Moeda =
          entidadeCloud.Moeda;
        acao.ValorAtualAcao =
          entidadeCloud.ValorAtualAcao;
        acao.ValorAnterior =
          entidadeCloud.ValorAnterior;
        acao.PercentualVariacao =
          entidadeCloud.PercentualVariacao;
        listResultado.Add(acao);
      }

      return listResultado;
    }
  }
}
```

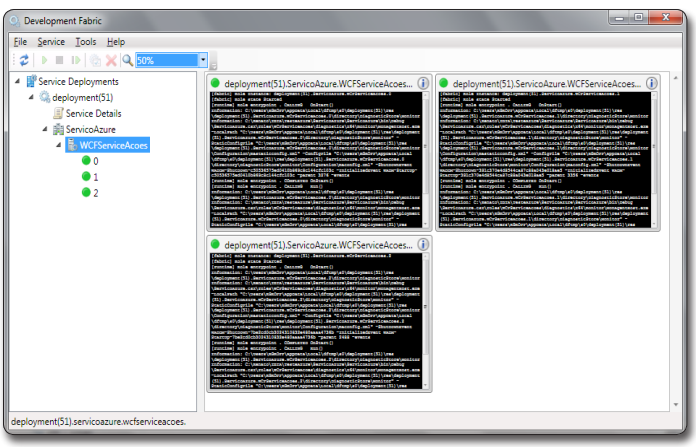


Figura 12. Role WCFServiceAcoes em execução no Development Fabric

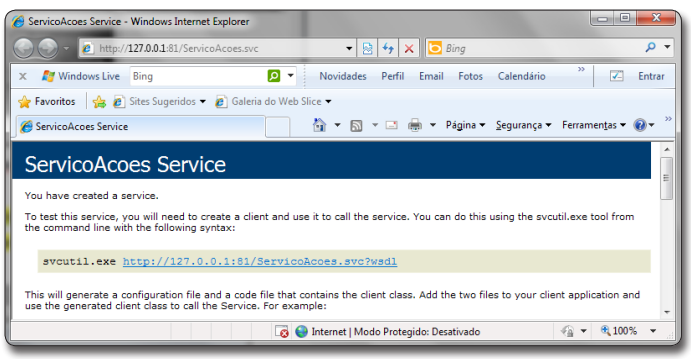


Figura 13. Acesso ao Web Service através do browser

ConsumidorServicoAzure será uma solução formada por um site on-premise, tendo sido criada com o objetivo de demonstrar a facilidade em se consumir um Web Service hospedado na "nuvem" (neste caso, através do ambiente de simulação oferecido pelo Development Fabric). A **Figura 14** apresenta a estrutura da mesma.

É imprescindível adicionar uma referência à *Web Role WCFServiceAcoes*, com o intuito de permitir que o serviço disponibilizado por ela seja consumido pela aplicação *ConsumidorServicoAzure*. Isto pode ser feito dentro do Visual Studio, clicando-se com o botão direito do mouse sobre o projeto e selecionando a opção *Add Service Reference* (**Figura 15**).

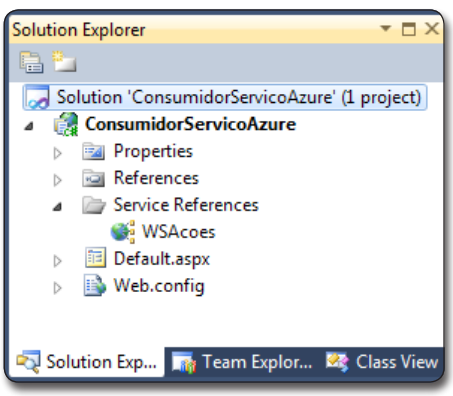


Figura 14. Estrutura da solução ConsumidorServicoAzure

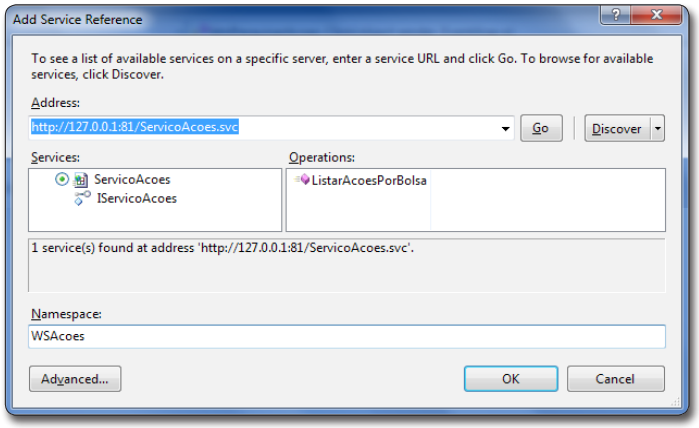


Figura 15. Adicionando uma referência ao Web Service na "nuvem"

A adição da referência à Web Role *WCFServiceAcoes* atualizará automaticamente o arquivo *Web.config*, da aplicação *ConsumidorServicoAzure*, gerando um item de configuração que aponta ao endereço de testes do Web Service, conforme indicado na **Listagem 10** (alguns itens de configuração foram omitidos, para efeitos de simplificação).

Listagem 10. Arquivo Web.config da aplicação ConsumidorServicoAzure.

```
<?xml version="1.0"?>
<configuration>
  ...
  <system.serviceModel>
    <bindings>
      <basicHttpBinding>
        <binding
          name="BasicHttpBinding_IServicoAcoes"
          ... />
          ...
        </binding>
      </basicHttpBinding>
    </bindings>
    <client>
      <endpoint
       address="http://127.0.0.1:81/ServicoAcoes.svc"
       binding="basicHttpBinding"
       bindingConfiguration=
         "BasicHttpBinding_IServicoAcoes"
       contract="WSAcoes.IServicoAcoes"
       name="BasicHttpBinding_IServicoAcoes"/>
    </client>
  </system.serviceModel>
</configuration>
```

A página *Default.aspx* (**Listagem 11**), por meio de sua funcionalidade de consulta às ações de uma determinada Bolsa de Valores, acessa o Web Service definido anteriormente neste artigo (e que está hospedado no Development Fabric).

Com a aplicação *ConsumidorServicoAzure* em execução, torna-se possível consumir o serviço *WCFServiceAcoes*, obtendo-se assim cotações para uma determinada Bolsa de Valores (**Figura 16**).

Listagem 11. Implementação da página Default.aspx.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace ConsumidorServicoAzure
{
  public partial class Default : System.Web.UI.Page
  {
    protected void btnPesquisarAcoes_Click(
      object sender, EventArgs e)
    {
      using (WSAcoes.ServicoAcoesClient servicoAcoes =
        new WSAcoes.ServicoAcoesClient())
      {
        grdDadosBolsaValores.DataSource =
          servicoAcoes.ListarAcoesPorBolsa(
            ddlBolsaValores.SelectedValue);
        grdDadosBolsaValores.DataBind();
      }
    }
  }
}
```

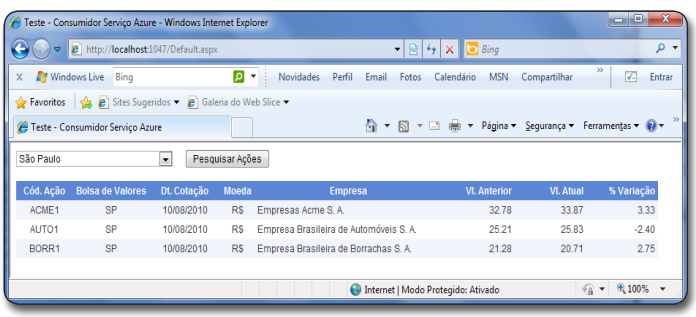


Figura 16. Site consumindo o Web Service de ações

Conclusão

O Azure, solução de *cloud computing* da Microsoft, oferece um conjunto variado de recursos, conforme apresentado neste artigo, buscando com isto ser a resposta a uma série de necessidades do ambiente empresarial.

A questão financeira pode representar um item importante a possíveis interessados nos componentes oferecidos pelo Azure, já que a cobrança se faz mediante o grau de utilização dos serviços. O que, por outro lado, vem de encontro à necessidade que muitas empresas possuem em reduzir investimentos, gastos com recursos em hardware, dando maior ênfase ao desenvolvimento de soluções específicas de software, essencial para o sucesso dos negócios.

Sua arquitetura, assim como a de outras plataformas do mesmo tipo, prima pelo fornecimento de serviços com alta disponibilidade, tolerância a falhas, escalabilidade e segurança.

Levando-se em conta todos estes fatores, bem como à sua estrutura, ao suporte dado a desenvolvedores pela Microsoft e à

utilização de padrões de comunicação de grande aceitação como REST, a plataforma Azure possibilita o desenvolvimento de aplicações robustas e dotadas de grande flexibilidade. Tais aspectos são, portanto, muito apreciados em um cenário corporativo de contínuas e aceleradas transformações.

Links

Artigo "Microsoft Case Study: Windows Azure – 3M"
http://www.microsoft.com/CaseStudies/Case_Study_Detail.aspx?casestudyid=4000005768

Artigo "Microsoft Case Study: Windows Azure – Kia Motors"
http://www.microsoft.com/casestudies/Case_Study_Detail.aspx?CaseStudyID=4000007550

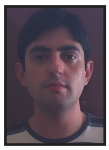
Artigo "Microsoft Case Study: Windows Azure – Siemens"
http://www.microsoft.com/casestudies/case_study_detail.aspx?casestudyid=4000005945

Cloud Computing - Locaweb
http://www.locaweb.com.br/solucoes/cloud-computing.html

Windows Azure Platform
http://www.microsoft.com/windowsazure/



Giuliano Paulo Velho Moreno
gpvmoreno@gmail.com
Atua no ramo de levantamento e especificação de requisitos de software há 8 anos. É Bacharel em Ciência da Computação, com especialização em Engenharia de Software.



Renato José Groffe
renato.groffe@yahoo.com.br
Atua como Consultor em atividades voltadas ao desenvolvimento de softwares há mais de 10 anos. Bacharel em Sistemas de Informação, com Especialização em Engenharia de Software. Microsoft Certified Technology Specialyst (Web, WCF, Distributed Applications, ADO.NET, Windows Forms), Oracle Certified Associate (PL/SQL), Sun Certified (SCJP, SCWCD), ITIL Foundation V2.



Tiago Valim
tvalim.ti@gmail.com
Atua no ramo de programação e desenvolvimento de softwares há 2 anos. É Tecnólogo em Processamento de Dados, com Especialização em Engenharia de Software.

DÊ UM SALTO EM CONHECIMENTO!



Acesse o maior portal para desenvolvedores da América Latina!

17 mil posts

280 mil cadastrados

5 milhões de page-views por mês

Os milagres do Jail no FreeBSD e os desafios da Virtualização

Denis Augusto

O FreeBSD é um sistema operacional nascido em dezembro de 1993, derivado do antigo BSD386. Com o evoluir de cada versão, diversas funcionalidades foram somadas, dentre elas pode-se citar:

• FreeBSD é um sistema operacional livre, do tipo *Unix-Like* e descendente do BSD, que por sua vez é outro sistema operacional UNIX, desenvolvido pela Universidade de Berkeley;

• Imensa quantidade de aplicações portadas de outros sistemas operacionais como Linux, NetBSD, OpenBSD, dentre outros (são mais de 26.000);

• Sistema de arquivos extremamente poderoso derivado do UFS (*Unix File System*) com *Soft Updates* (UFS2). O *Soft Updates* foi introduzido no FreeBSD por Marshall Kirk McKusick e possui diversas qualidades, dentre elas uma poderosa verificação de integridade do sistema de arquivos, sendo uma ótima alternativa ao sistema de arquivos *journaling* (veja a **Nota DevMan 1**). Além disto, em conjunto com os recursos contidos no UFS2, tem-se um bom tratamento de grandes volumes de dados e I/O de disco;



Nota do DevMan 1

Journaling

De acordo com a Wikipédia, um sistema de arquivos com journaling possibilita ao Sistema Operacional manter um log (journal) de todas as mudanças no sistema de arquivos antes de escrever os dados no disco. A partir disso, caso ocorra algum erro durante tais mudanças, é possível restaurar a consistência do sistema de arquivos sem a necessidade de verificar cada arquivo.

• Cuidado extremo com a compatibilidade dos fabricantes de hardware mais conhecidos, aprimorando os *drivers* para uma melhor comunicação com os hardwares especificamente desenvolvidos para servidores. Isso fornece uma segurança para quem usa o FreeBSD e aprimora a HAC (*Hardware Access List*/Lista de Hardwares Acessíveis);

• Apresenta segurança robusta com o filtro de pacotes PF (*Packet Filter*), originado do OpenBSD, um sistema operacional específico para a tarefa de *Firewall*;

• Possibilita o uso de QoS (*Quality of Service*), CARP (*Common Address Redundancy Protocol*), VLAN (*Virtual Local Area Network*) e *link aggregation* já integrados ao *kernel* do sistema operacional;

• Faz de forma simples o balanceamento de *gateways* com recursos incorporados ao *kernel*;

• Avaliando a licença BSD, observa-se flexibilidade para a composição de soluções comerciais. Um exemplo é o sistema operacional para *firewall* IPSO, desenvolvido pela Nokia para os equipamentos *Check Point*, baseando-se totalmente no FreeBSD. Esse, sem dúvida, é um exemplo comercial de sucesso usando FreeBSD.

Resumo DevMan

De que se trata o artigo:

Aborda como construir uma solução segura que eleve a proteção de um projeto que tem como meta publicar um servidor para acessos de qualquer usuário na Internet. Para isso, será utilizado o Jail, que é um recurso de criação de múltiplos ambientes virtuais nativo do FreeBSD, como base de um processo de virtualização.

Para que serve:

Auxilia de forma teórica e prática como compor uma solução de Jail usando o sistema operacional FreeBSD como uma opção de hardening. Hardening é um processo de mapeamento das ameaças, mitigando os riscos e executando atividades de correção. Possui foco na infraestrutura e o objetivo principal é prepará-la para enfrentar tentativas de ataque.

Em que situação o tema é útil:

Quando se tem um servidor e deseja-se aproveitá-lo seguindo o conceito de virtualização de sistemas operacionais FreeBSD, usando somente recursos open source nativos do sistema operacional.

Observando estas funcionalidades é simples entender os motivos que fazem com que diversas empresas façam uso do FreeBSD, dentre elas: Yahoo!, Apache, Pair Networks, Blue Mountain Arts, Google, Fotolog, Orkut, MySpace, YouTube, Second Life e muitas outras. Analisando este cenário no Brasil, a adoção do FreeBSD vem crescendo com o tempo, sendo usado em diversas entidades governamentais, universidades e empresas.

São diversos exemplos de sucesso que fazem uso do FreeBSD, e este artigo aborda uma dessas possibilidades com Jail, uma metodologia poderosa, muito usada no passado e que ainda faz sucesso dentre muitos gerentes de rede.

O que é Jail e quais seus benefícios?

Primeiro precisa-se saber que Jail é um conceito antigo, usado por todos os ambientes Unix, e *Unix-Like* (que nada mais são do que sistemas operacionais parecidos com o Unix), como uma alternativa segura de publicar uma aplicação para acesso externo, isolando-a do resto do sistema operacional. Entretanto, Jail no FreeBSD representa uma evolução desse conceito, isolando não

só uma aplicação, mas um sistema operacional por completo, com pilhas TCP/IP individuais, áreas de memória e acessos a disco, representando uma opção para a virtualização de sistemas operacionais FreeBSD.

Este texto revela como compor um ambiente em Jail no FreeBSD usando exemplos práticos, e também analisa seu uso aplicado à computação em nuvem e outras opções para a virtualização de sistemas operacionais.

É comum confundir Jail com outra metodologia muito presente no passado do universo UNIX e ainda usada até hoje, conhecida como *Chroot*. Deve-se ter em mente que *Chroot* (**Figura 1**) e Jail (**Figura 2**) são metodologias diferentes, com aplicabilidades bem específicas. Nota-se que o *Chroot* cria um novo diretório raiz (diretório virtual) e limita a aplicação a este diretório. Todas as bibliotecas, áreas de disco e demais acessos devem estar dentro deste novo diretório raiz. Esse método ainda é usado por aplicações tradicionais como o BIND (*Berkeley Internet Name Domain*) para a resolução de DNS de forma mais segura, pois eleva e muito o nível de segurança, isolando o raio de ação no momento de um ataque.

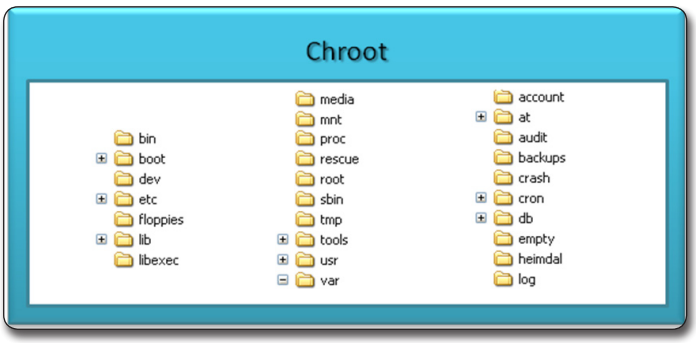


Figura 1. Funcionamento do Chroot (diretórios, arquivos e bibliotecas)

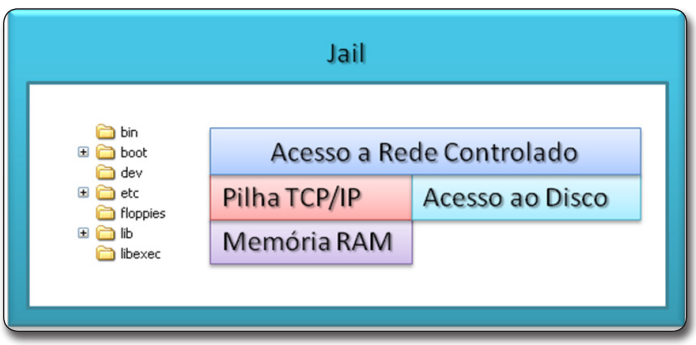


Figura 2. Funcionamento do Jail (maior controle das áreas do sistema operacional)

Comparando o Jail com o Chroot, observa-se que o segundo não separa a pilha TCP/IP ou controla os acessos de forma tão rígida como o Jail. Quando se usa Jail, o *kernel* do sistema operacional individualiza as áreas de rede, memória, processos, usuários e arquivos. Além disso, existem variáveis de *kernel* específicas para controlar as atividades de um ambiente Jail. Cada Jail pode interagir com outros ambientes da mesma forma que um ambiente computacional tradicional, diferenciando-se pela maior segurança a ele aplicada. O Jail cria um verdadeiro *container*, o que abre um leque de possibilidades para um ambiente de produção, homologação ou desenvolvimento mais seguro usando como se fosse um clone do sistema operacional principal aplicado a outra atividade, com um novo endereço IP, pilha TCP/IP e área de memória.

Do ponto de visa da segurança, tanto o Chroot quanto o Jail criam áreas de maior isolamento e elaboram a proteção da aplicação separando o diretório visto pela aplicação do sistema operacional, criando o que pode ser visto como uma virtualização de diretórios. Se o ambiente for atacado com sucesso, o atacante verá uma área isolada, ficando com sua ação limitada. Todas as informações publicadas dentro desta área estarão isoladas do ambiente principal, sendo muitas destas informações irrelevantes para o funcionamento do Jail como os arquivos de senhas, por exemplo. Assim, visando elevar a proteção do ambiente principal, recomenda-se depositar informações não verdadeiras para confundir o atacante. Como os *backups* são baseados em áreas de diretórios, basta retornar uma cópia de um período antes do ataque para ter o ambiente funcional novamente e estudar os motivos que deixaram o sistema vulnerável.

A primeira versão de Jail foi introduzida no FreeBSD 5.1. Depois desta data diversas modificações foram realizadas e hoje se pode contar com funcionalidades como IPv6, múltiplos endereços IP e com ambientes contendo binários de 32bits em um sistema operacional host de 64bits. Nas seções seguintes será visto como é simples configurar e gerenciar um ambiente em Jail com o uso do FreeBSD.

Desvendando um ambiente em Jail no FreeBSD

Quando se trabalha com Jail no FreeBSD, faz-se uso de ferramentas inclusas no sistema operacional e de um *kernel* preparado para este tipo de operação, dispondo sempre de um manuseio muito simplificado. Pode-se dizer que em toda a operação observam-se quatro ferramentas (três aplicativos e um *script*): *jexec*, *jls*, *jail* e do script */etc/rc.d/jail*. O primeiro entra na console do ambiente Jail, o segundo lista os ambientes existentes, o terceiro cria o Jail e o último prepara os ambientes Jail de forma automatizada após o *boot* do sistema *host*.

Quando um ambiente em Jail é criado, pode-se fazê-lo da forma simples (automatizada) ou da forma mais trabalhosa (manual). Se for desejada a construção de um ambiente da forma manual, todas as bibliotecas e diretórios necessários para o funcionamento da aplicação deverão ser instalados manualmente. Para isto, cada comportamento da aplicação (necessidade de bibliotecas, diretórios para depósito de arquivos de *logs*, diretórios para a leitura de arquivos de configuração, etc.) tem que ser mapeado. A esse mapeamento define-se como *dependência*. A verificação de bibliotecas necessárias pode ser simplificada com o comando *ldd*. Um exemplo clássico para se entender esse mapeamento pode ser visto com o arquivo binário do jogo Doom (**Listagem 1**).

Observa-se que os diretórios */usr/X11/lib* e */lib* devem existir no ambiente em Jail e estas bibliotecas (*libXt.so.3.1.0*, *libX11.so.3.1.0* e *libc.so.4.6.29*) devem ser copiadas neles exatamente no mesmo diretório. Sem dúvida que esse é um ponto extremamente trabalhoso para um processo manual, pois se pode encontrar uma

aplicação que possui muitas dependências. Assim, o melhor caminho para criar o ambiente Jail é fazer a instalação do novo ambiente com o CD/DVD e, se necessário for, pode-se remover o que não é preciso (diretórios, arquivos, dispositivos de acesso, bibliotecas). Observa-se agora a **Listagem 2**, que inicia o processo de criação do Jail com a definição do diretório onde residirá toda a estrutura do mesmo.

O parâmetro *–p* faz com que toda a árvore de diretório seja criada caso não exista. Foi usado o diretório *"/usr"* porque, por padrão, esse é o diretório que tem mais espaço livre, mas qualquer diretório pode servir como base de um Jail.

Para que o Jail funcione, precisa-se de mais um endereço IP (*IP alias*) na interface primária do FreeBSD, e isso pode ser feito via linha de comando ou de forma definitiva modificando o arquivo de inicialização. Assim, é necessário modificar o arquivo */etc/rc.conf* colocando os *IP alias* necessários, conforme a **Listagem 3**.

Definidas as informações do *alias*, pode-se ativá-lo e reconfigurar o ambiente que fornecerá os acessos à rede local ou à Internet ao Jail (**Listagem 4**).

Listagem 1. Verificando dependência de bibliotecas.

```
# ldd linuxxdoom
libXt.so.3 (DLL Jump 3.1) => /usr/X11/lib/libXt.so.3.1.0
libX11.so.3 (DLL Jump 3.1) => /usr/X11/lib/libX11.so.3.1.0
libc.so.4 (DLL Jump 4.5pl26) => /lib/libc.so.4.6.29
#
```

Listagem 2. Criando diretório para o Jail.

```
# mkdir –p /usr/jail/Jail_Exemplo1
```

Listagem 3. Criando o IP alias.

```
ifconfig_rl0_alias0="inet 10.10.10.120 netmask 255.255.255.0"
```

Listagem 4. Aplicando a nova configuração de rede.

```
# sh /etc/netstart
Devd already running? (pid=3001)
rl0: flags=10289(UP,BROADCAST,RUNNING, SIMPLEX, MULTICAST, NEEDSGIANT>
mtu 1500
   inet6 fe80::20c:29aa:feda:8449%rl0 prefixlen 64 scopeid 0x1
   inet 10.10.10.120 netmask 0xffffff00 broadcast 10.10.10.255 ether
00:0c:29:ab:84:49
add net default: gateway 10.10.10.1
Additional routing options:.
#
```

Assim, a interface física Realtek (*rl0*), previamente configurada com o endereço IP 10.10.10.120 e máscara 255.255.255.0 (24 bits), receberá mais um endereço para trabalho (10.10.10.120). Importante deixar claro que um *alias* pode ou não estar na mesma *subnet* do endereço primário.

Tendo os requisitos de diretório e endereço IP necessários para a operação do Jail, basta seguir com a instalação do sistema operacional. Aqui será utilizado um DVD da versão 8.0 (**Listagem 5**).

Tomando como base o aspecto segurança, é indicado fazer uso de pacotes previamente compilados (*packages*) ou compilar a aplicação e depois remover os diretórios produtos desta compilação (**Listagem 6**).

Podem-se depositar no diretório do Jail todos os *devices* (dispositivos de acesso como a unidade de DVD ou partição do disco rígido) necessários ao seu funcionamento. Isto pode ser feito de forma simples com o uso do comando *cp*, de acordo com a **Listagem 7**.

Nunca devem ser usados *devices* que controlem o acesso ao disco ou o acesso direto aos *inodes* (veja a **Nota DevMan 2**), pois existem métodos de invasão específicos que possibilitam o acesso ao sistema *host* via *inode*. Desta forma, deve-se deixar somente o necessário, removendo compiladores, área para compilação de fontes e comandos não usados como *tftp*, *ftp* e *telnet*. Isto é válido não só para o ambiente em Jail, mas para servidores tradicionais.



Nota do DevMan 2

Inode

Inode é uma estrutura de dado fisicamente posicionada no disco rígido para guardar arquivos, diretórios ou qualquer outro objeto do sistema operacional.

Existem casos em que a aplicação existente no ambiente Jail requisita para o seu funcionamento o diretório */dev* exatamente como o ambiente *host*. Nestes casos, é indicado fazer o que exibe a **Listagem 8**.

Listagem 5. Instalando o sistema operacional no Jail.

```
# mkdir /mnt/dvdrom
# mount –t cd9660 /dev/acd0 /mnt/dvdrom
# cd /mnt/dvdrom/8.0-RELEASE/base
# cat base.?? | tar --unlink -xpzf - -C /usr/jail/Jail_Exemplo1
```

Listagem 6. Customizando a estrutura de file system do Jail.

```
# mount_nullfs /usr/ports /usr/jail/Jail_Exemplo1
# mount_nullfs /usr/src /usr/jail/Jail_Exemplo1
```

Listagem 7. Criando a estrutura de dispositivos.

```
# cp -R /dev/null /usr/jail/Jail_Exemplo1/dev
# chmod 666 /usr/jail/Jail_Exemplo1/dev/null
# cp -R /usr/jail/Jail_Exemplo1_web/dev
# chmod 666 /usr/jail/Jail_Exemplo1/dev/zero
```

Listagem 8. Montando o diretório /dev do host.

```
# mount -t devfs devfs /jail/Jail_Exemplo1_web/dev
```

O próximo passo é criar manualmente o Jail (**Listagem 9**). Essa tarefa pode ser automatizada com o *boot* (iniciar) do sistema operacional *host*, e isto será visto em detalhes neste artigo.

Na **Listagem 9** pode-se ver que a aplicação Jail é usada com diversos parâmetros, onde o primeiro indica o caminho (*Path*) que contém a estrutura de diretórios, bibliotecas e programas necessários ao sistema operacional dentro do Jail. O segundo parâmetro

é o FQDN (*Fully Qualified Domain Name*) do servidor, seguido de seu endereço IP (*alias* da interface, definido anteriormente) e o ambiente *shell* para gerenciamento em linha de comando.

Como boa prática, é indicada a criação dos arquivos */etc/fstab* (arquivo vazio) e */etc/resolv.conf* dentro do ambiente Jail. Isso é necessário para um melhor funcionamento de aplicações que podem ser necessárias dentro do Jail.

Até o momento foi apresentado como criar manualmente um ambiente em Jail, mas pode-se automatizar este processo fazendo uso do script */etc/rc.d/jail*. Para isso, é necessário inserir todas as informações necessárias no arquivo */etc/rc.conf* (**Listagem 10**).

Listagem 9. Criando o Jail.

```
# jail /usr/jail/Jail_Exemplo1 servidor01.dominio.com.br 10.10.10.120 /bin/csh
```

Listagem 10. Automatização do Jail no processo de boot.

```
# Deseja ativar o uso de Jails?
jail_enable="YES"
#
# Qual o nome do jail que será criado?
jail_list="servidor01"
#
# Decida se o jail trabalha somente com o protocolo TCP/IP
jail_socket_unixiproute_only="YES"
#
# O usuário root no jail pode mudar o nome do ambiente?
jail_set_hostname_allow="NO"
#
# Vai montar o devfs no ambiente Jail?
jail_servidor01_devfs_enable="YES"
#
# Vai montar o fdescfs no ambiente Jail?
jail_servidor01_fdescfs_enable="NO"
#
# Decida se vai permitir o uso de SystemV IPC
jail_sysvipc_allow="NO"
#
# Será montado o procfs no ambiente Jail?
jail_servidor01_procfs_enable="YES"
#
# Qual será o hostname no ambiente Jail?
jail_servidor01_hostname="servidor01.dominio.com.br"
#
# Qual será o diretório root do Jail?
jail_servidor01_rootdir="/usr/jail/Jail_Exemplo1"
#
# Qual será o endereço IP?
jail_servidor01_ip="10.10.10.120"
#
# Podemos executar um script quando iniciar o ambiente Jail
jail_servidor01_exec_start="/bin/sh /etc/rc"
#
# Podemos definir o comando para executar quando parar o ambiente Jail
jail_servidor01_exec_stop="/bin/sh /etc/rc.shutdown"
#
# Permitirá montar e desmontar sistema de arquivos?
jail_servidor01_mount_enable="NO"
```

Com estas informações pode-se colocar o ambiente do Jail em funcionamento usando o script:

```
# /etc/rc.d/jail start
```

Para deixar este ambiente inoperante, basta fazer:

```
# /etc/rc.d/jail stop
```

As variáveis do *kernel* que gerenciam a segurança de um ambiente Jail podem ser verificadas com o comando *sysctl*:

```
# sysctl –a | grep jail
```

Na **Listagem 11** é possível verificar como exibir os ambientes Jail que estão em funcionamento no host, através do comando *jls*.

Observa-se que este Jail tem a identificação (ID) 1 e as demais informações de IP, FQDN (*Fully Qualified Domain Name*) e diretório *Path* também são exibidas. Deste modo, pode-se entrar na console administrativa do Jail executando um ambiente *shell* desejado (*csh*), como exibido linha seguinte:

```
# jexec 1 csh
```

Dentro do ambiente Jail, podem-se fazer diversas atividades; dentre elas, parar suas operações sem fazer uso do script */etc/rc.d/jail*, bastando usar o comando *kill*:

```
# jexec 1 csh
# kill –KILL 1
```

Como o Jail isola o novo sistema operacional em uma área completamente diferente, pode-se, com isso, instalar aplicativos diretamente da console do ambiente Jail. Como exemplo, pode-se verificar a instalação do MySQL via *packages* previamente depositados no Jail (**Listagem 12**).

Listagem 11. Exibindo os ambientes Jails em operação.

```
# jls
JID IP Address Hostname Path
1 10.10.10.120 servidor01.dominio.com.br /usr/jail/Jail_Exemplo1
```

Listagem 12. Instalação do MySQL no Jail.

```
# jexec 1 csh
# pkg_add msql41-server
# pkg_add mysql41-client
```

Usando Jail na Nuvem

O termo computação em nuvem, ou *cloud computing*, não é um tema novo, mas ainda agita muito as rodas de discussão tecnológicas. Fazendo uma análise macro, pode-se notar que ele nada mais é do que a forma mais completa de virtualização de um ambiente.

Observa-se que o desenho da nuvem é composto por um ou mais *pools* de servidores físicos equipados adequadamente com software de virtualização, compartilhando recursos como *storage*, *switches*, energia e meio físico no *Data Center*, fazendo uso de controle e gerenciamento centralizado.

Além disso, quando se imagina os sistemas operacionais sendo virtualizados neste ambiente, vê-se a possibilidade de poder usá-lo

em qualquer um desses servidores físicos, melhorando o SLA (*Service Level Agreement* - Acordo de Nível de Serviço) do serviço entregue, como escalabilidade, versatilidade em duplicação de ambientes, *failover*, segurança, eficiência em uso de recursos de processador e memória, e com um melhor aproveitamento energético. Se isto tudo não for suficiente, ainda pode-se dizer que a virtualização, esteja ela compondo um ambiente em nuvem ou não, está contribuindo significantemente para a chamada *Green IT*, conservando recursos ambientais.

Computação em nuvens difere-se de diversos ambientes e não se pode confundi-la com a computação em *grid*, que é a combinação de computadores em *cluster* combinados para uma função em comum, compartilhando recursos como processamento e memória. Um exemplo clássico em banco de dados é o uso do Oracle RAC (*Real Application Clusters*), que fornece um *cluster* de vários nós Ativo-Ativo e compartilha memória RAM entre os nós do *cluster*.

Ainda pensando na computação em nuvem, pode-se dizer que existem diversos tipos, mas este artigo irá centrar os estudos em apenas três deles: pública (*public*) ou externa, privada (*private*) ou interna e híbrida (*hybrid*).

Para entender a diferença entre elas, imagine que na privada o compartilhamento de recursos é feito dentro de uma rede privada de dados específica para uma corporação, enquanto a pública é a forma mais tradicional, compartilhando recursos entre as corporações, com provisionamento dinâmico entre Data Centers. Já o modelo híbrido se posiciona como um modelo onde a corporação pode usufruir dos benefícios e estar tanto na pública quanto na privada.

Um conceito que ainda gera discussões é o uso de estruturas chamadas de *interclouds* (internuvens), onde se tem o relacionamento entre nuvens destinadas a diferentes organizações ou uma interconexão global de nuvens posicionadas em locais diferentes do planeta. Com isso, as corporações podem ter um nível de interação maior ou até ativar/transferir recursos de uma nuvem para a outra.

A pergunta que se deve ter em mente é: será que computação em nuvem é tão boa assim? Será que se aplica a todos os casos ou existem limitações? Onde se podem ter problemas inviabilizando o uso da computação em nuvem? A resposta é simples, onde existem normativos internos das corporações deixando claro o não trânsito de informações inerentes à corporação entre *Data Centers* ou onde existe a proibição de estruturas virtualizadas. Para os casos onde são exigidos componentes dedicados, basta projetar servidores específicos para o uso somente da organização em questão. O fato é que diversas qualidades e facilidades podem ser encontradas no que prega a computação em nuvem. Ela é uma tendência irreversível, e as empresas têm que se adaptar a ela.

Agora que já foi definido o que é a computação em nuvem, pode-se aplicar esse conhecimento para a tecnologia de virtualização de Jails com FreeBSD. Para isso, será referenciado o modelo de uma nuvem privada (*private cloud computing*) em um grupo de servidores físicos executando um software para a virtualização de sistemas operacionais como o VMware, Hyper-V, VirtualBox e Xen, por exemplo. Nesta estrutura serão instalados três servidores FreeBSD 8.1 chamados de Marte, Jupiter e Saturno (**Figura 3**).

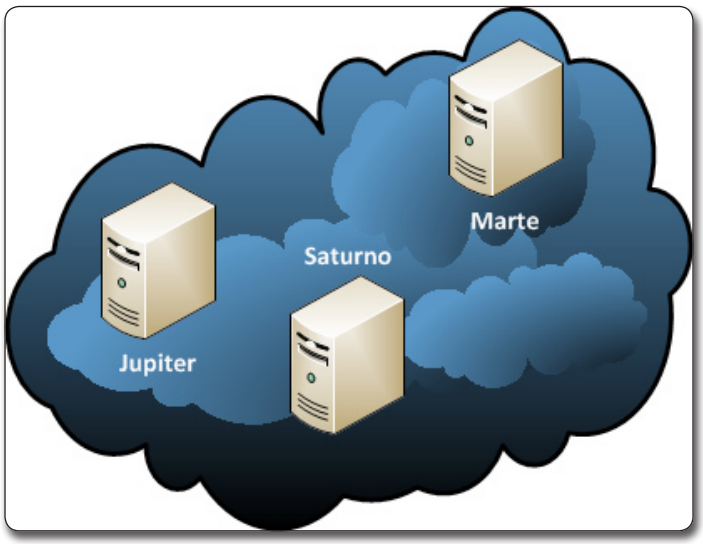


Figura 3. Instalando o FreeBSD na private cloud computing

Podem-se criar quantos Jails forem necessários, e para o nosso exemplo, serão configurados dois Jails no servidor Marte e um Jail no servidor Jupiter. Por fim, no servidor Saturno serão configurados mais dois Jails (**Figura 4**).

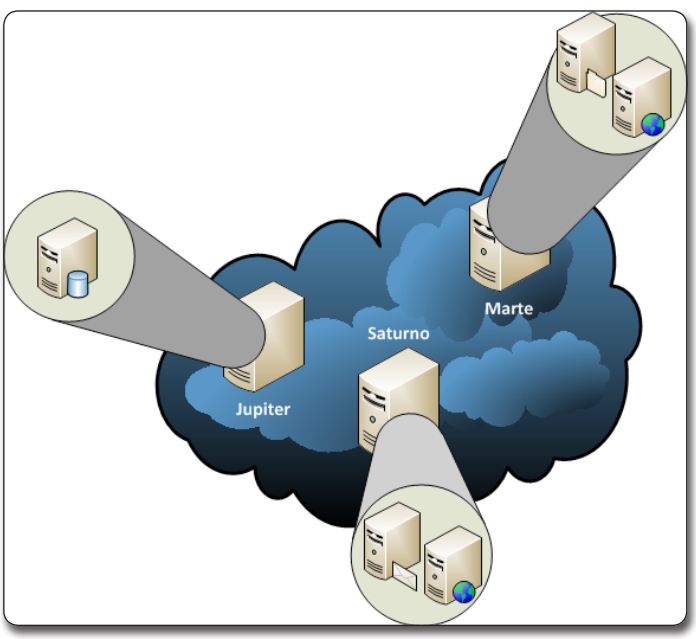


Figura 4. Configurando os ambientes em Jail

Fazendo uma análise mais técnica, pode-se notar uma camada de virtualização, distribuída entre um *pool* de servidores formando uma computação em nuvem. Percebe-se que só estão sendo contratadas três máquinas, mas na verdade têm-se presentes oito servidores. Isso proporciona economia para o projeto na área de

Data Center contratada e tira um melhor proveito do hardware físico usado para os servidores, sem falar que torna o projeto aderente a *Green IT* (TI Verde), outra tendência de mercado.

Comparações

O modelo de Jail do FreeBSD remete a fazer algumas comparações com conceitos aplicados em ambientes similares aos existentes em sistemas operacionais bem respeitados e conhecidos como o Solaris/Open Solaris, AIX e os próprios IBM Mainframes.

Buscando os registros históricos, vê-se que todo o processo de virtualização se iniciou com os *mainframes*, pois eram máquinas (e ainda são) muito caras, e precisava-se justificar de forma concreta sua compra para a aplicação de diversas tarefas. Assim, a IBM criou mecanismos para individualizar áreas dentro dos *mainframes* para serem criados diversos servidores com funções independentes. Tais áreas são chamadas de *containers* e podem ser encontradas nos atuais sistemas operacionais IBM System z (IBM z/OS) com o uso de virtualizações baseadas no *hypervisor* (IBM z/VM).

Analisando de forma simples, quando se usa virtualização, deseja-se reutilizar o hardware físico e compartilhá-lo com outros sistemas operacionais visando um aproveitamento máximo de recursos sem prejudicar o desempenho. Isto ocorre com diversas tecnologias que evoluíram, saindo do mundo *mainframe* e chegando até o universo Unix, para depois ser aplicada aos computadores Intel/AMD x86. São exemplos disto o VMware, Xen, Microsoft Hyper-V (**Figura 5**) e tantos outros.

O elemento responsável por exercitar a virtualização é chamado de *Hypervisor* e pode fazer ou não o papel de um sistema operacional equipado com as ferramentas necessárias para aplicar a virtualização e, como tal, sofre impactos como qualquer sistema operacional (carga de processamento, elevação do volume de requisições a memória RAM, consultas a áreas do disco feitas com mais frequência, etc.). Da mesma forma, o Jail aplicado no FreeBSD sofrerá o mesmo tipo de impacto e, sem dúvida, em qualquer projeto essas questões devem e têm que ser consideradas para que o usuário final da aplicação não seja penalizado com lentidões ou perdas de desempenho.

Assim, como se resolve essa questão? Basta, para isso, adequar o projeto na fase de concepção aos requisitos de carga da aplicação e especificar o hardware mais adequado. O problema é quando não se sabe que carga é essa, tendo, nestes casos, que contar com bom senso e um NOC (*Network Operations Center* - Centro de Operações de Rede) muito bem treinado para monitorar e indicar pontos de gargalos para a área corporativa de SLM (*Service Level Management*). Uma área corporativa de SLM é um conceito bem discutido no ITIL (*IT Infrastructure Library*), e é um modo prático de se antever aos incidentes e problemas.

Retornando ao modelo de virtualização aplicado nos Mainframes IBM, que foi introduzido em 1967, com o z/VM *hypervisor*, vê-se que é um funcionamento um pouco diferente dos modelos aplicados em plataformas x86. Observa-se que existe também uma granularidade customizada para melhor aproveitamento dos recursos de memória, processador, *storage* e rede de dados. Isso possibilita a virtualização de uma infinidade de máquinas em um único equipamento, usufruindo de um hardware que fornece um SLA (*Service Level Agreement*) extremamente elevado, com diversos níveis de redundância e suporte avançado. Pensando ainda em fácil crescimento, é prática equipar o hardware com todos os elementos, como CPU, memória e licenciá-los conforme a necessidade de crescimento, assim, facilmente com um telefonema pode-se adicionar mais desempenho ao ambiente de forma simples e rápida.

Em 1989, este modelo de virtualização foi aprimorado com o advento da tecnologia LPAR (*Logical Partition*s – Partições Lógicas). O LPAR enquadra-se como um elemento de virtualização no nível do hardware e pode fornecer um elevado nível de abstração aliado a um grau de segurança aplicado a mais de 60 *containers* em um único equipamento.

Dessa forma, pode-se particionar uma máquina física em vários LPARs, cada um hospedando um sistema operacional distinto, compartilhando os mesmos recursos de hardware. A garantia deste compartilhamento é feita por um elemento chamado de PR/SM (*Processor Resource/System Manager*), que se integra com o IBM System z e representa para os ambientes IBM o *hypervisor* que é executado diretamente no nível do hardware.

Hoje LPARs podem ser encontrados em equipamentos não *mainframe*, como as máquinas IBM pSeries e IBM iSeries, criando múltiplas instâncias em um único hardware. Os LPARs permitem a segurança necessária para virtualizar bases de dados, um dos calcanhares de qualquer ambiente de virtualização, e acredita-se que só se estará livre desta problemática quando o próprio *hypervisor* – por software – garantir os IOPS (I/O por segundo) necessários para a validação deste projeto de virtualização.

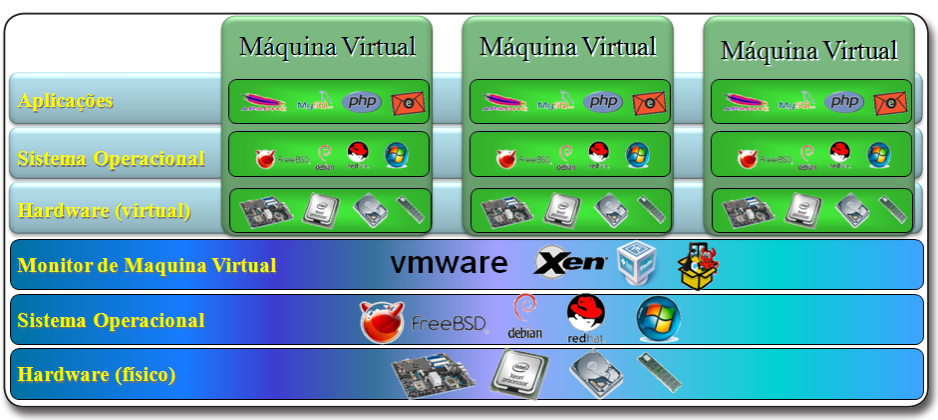


Figura 5. Estrutura de um ambiente de virtualização

Outro desafio que um ambiente virtualizado enfrenta é o licenciamento de alguns softwares, principalmente se o serviço é ofertado a terceiros. Para plataformas Microsoft isso tem que ser tratado com muito cuidado com seu gerente de parceria. Para bancos de dados, o que se deve ter em mente é se vai ser perdido o suporte contratado para o ambiente em questão ou se serão encontradas limitações para o atendimento, ou até se será garantido os IOPS (*Input/Output Operations Per Second*) desejados pelo projeto. Para softwares aplicados a ambientes ERP (*Enterprise Resource Planning*) ou SIGE (Sistemas Integrados de Gestão Empresarial) a necessidade é verificar se o mesmo está homologado para operar em ambiente de virtualização. Todos estes questionamentos devem ser feitos no início do projeto para que não sejam cometidos erros com impactos graves na precificação ou na própria invalidação da solução. A precificação é uma tarefa que deve ser feita na fase de projeto levando em conta todos os recursos necessários para o andamento do projeto.

Conclusão

Quando se pensa em Jail aplicado a um sistema equipado com FreeBSD, ou qualquer outra modalidade de virtualização, imagina-se um universo de possibilidades com áreas de memória distintas para cada Jail equipadas com portas TCP/IP e endereços IP exclusivos para cada área, além de uma poderosa ferramenta para *virtualização* de sistemas operacionais sob o mesmo ambiente do sistema *host* (hardware principal).

Isso se deve não só às bases sólidas herdadas do Jail existente em plataformas Sun Solaris, mas pela segurança, facilidade no manuseio e na configuração de um ambiente desse porte no FreeBSD.

Assim, sem nenhuma dúvida, se o sistema operacional usado for FreeBSD e se é desejada a composição de vários servidores FreeBSD, usando a mesma versão ou não da máquina *host*, a melhor opção é fazer uso do Jail, mesmo tendo opções mais amplas e mais poderosas como o VirtualBox. Mas falar de VirtualBox é outra história.



Denis Augusto
denis.augusto@gmail.com
Especialista em segurança de redes, projetista de soluções para automatização de tarefas e auditor de segurança para ambiente de servidores, autor do livro FreeBSD: O Poder dos Servidores em Suas Mãos (Editora Novatec), formado em Segurança de Dados pela Academia Latino-Americana de Segurança da Informação, especialista em integração de sistemas heterogêneos, unindo ambientes Windows e Linux, certificado em Microsoft Certified Systems Engineer- Security (MCSE:Security), Microsoft Certified Systems Administrator – Security (MCSA:Security), MCSA:Messaging, GIAC Cutting Edge Hacking Techniques Certificate, ITIL Foundation. Possui trabalhos publicados na área de segurança em eventos como SSI (Simpósio sobre Segurança em Informática em São José dos Campos, São Paulo), ICIE (International Congress on Informatics Engineering em Portugal, Lisboa), Latin American Conference of Informatics CLEI (Quito-Equador) e InfoNordeste. É instrutor da 4LINUX e criador dos cursos de FreeBSD e OpenBSD.

Links

FreeBSD Brasil
http://www.freebsdbrasil.com.br/

Why FreeBSD
http://www.ibm.com/developerworks/opensource/library/os-freebsd/

Virtualization Strategies Architectural Overview
http://www-01.ibm.com/redbooks/community/display/REDP4480/Virtualization+Strategies+Architectural+Overview

Conhecimento faz diferença!



Faça já sua assinatura digital ! | www.devmedia.com.br/es

Faça um *upgrade* em sua carreira

Em um mercado cada vez mais focado em qualidade, ter conhecimentos aprofundados sobre requisitos, metodologia, análises, testes, entre ouutros, pode ser a diferença entre conquistar ou não uma boa posição profissional. Sabendo disso a DevMedia lançou uma publicação totalmente especializada em Engenharia de Software. Todos os meses você pode encontrar artigos sobre Metodologias Ágeis; Metodologias tradicionais (document driven); ALM (application lifecycle); SOA (aplicações orientadas a serviços); Análise de sistemas; Modelagem; Métricas; Orientação à Objetos; UML; testes e muito mais. **Assine Já!**

DevMedia

Desafios da telefonia IP

Conheça as diferenças entre a telefonia IP e a voz sobre IP

André Koide da Silva

Resumo DevMan

De que se trata o artigo:

Neste artigo serão apresentadas as diferenças entre a telefonia IP e a voz sobre IP, além das características e as vantagens destas tecnologias. Também serão descritos os desafios e as limitações impostas pela utilização das redes de dados IP para o transporte da voz digitalizada. Por fim, serão expostas algumas soluções comerciais atualmente disponíveis no mercado, destacando suas funcionalidades principais e custos.

Para que serve:

As tecnologias de voz sobre IP estão em crescente expansão e popularização. Assim, este artigo procura esclarecer as diferenças, características e vantagens de sua utilização e implementação. Com isso, os profissionais e demais usuários da área de Tecnologia da Informação poderão compreender melhor como integrar e extrair os benefícios da adoção destas soluções em suas empresas.

Em que situação o tema é útil:

Os conceitos apresentados têm por objetivo fornecer a fundamentação teórica básica para que profissionais e usuários compreendam as características principais das soluções de voz sobre IP, possibilitando a escolha e a implementação do produto mais adequado às suas necessidades e de suas empresas, aproveitando todos os benefícios destas novas tecnologias.

A popularização e a disponibilização crescente de acessos banda larga aos usuários domésticos, bem como a redução nos custos para contratação de enlaces privativos junto às operadoras de telecomunicações (mercado corporativo) e o aumento da capilaridade da infraestrutura das redes de dados IP têm muito contribuído para expansão das tecnologias de telefonia IP e voz sobre IP.

Neste sentido, neste artigo serão abordadas inicialmente as diferenças entre estas tecnologias, pois ainda que ambas utilizem as funcionalidades do protocolo IP, existem importantes particularidades em suas características. Também serão destacadas as vantagens da adoção da telefonia IP e da voz sobre IP em relação aos sistemas de telefonia convencionais. A seguir, serão descritos os principais modelos de implementação da telefonia IP, analisados conforme o nível de resiliência desejado (redundância e segurança), e os desafios da conversão da voz em sinais digitais – formato utilizado para transmissão em redes de dados IP. Finalmente, será exposta uma tabela comparativa com as principais características e custos de algumas soluções atualmente disponíveis no mercado.

Telefonia IP versus VoIP (voz sobre IP)

Uma dúvida muito comum, mesmo entre os profissionais da área de tecnologia da informação, é sobre a diferença entre telefonia IP e voz sobre IP (VoIP). Muitas vezes estes termos são empregados como sinônimos. Embora ambas as tecnologias utilizem as funcionalidades do protocolo IP (Internet Protocol), suas características são substancialmente diferentes.

Os sistemas VoIP são aqueles que convertem o sinal analógico, gerado normalmente por sistemas de telefonia convencional, para um sinal digital (sequência de zeros e uns) que é transportado pelo protocolo IP. Este, hoje em dia, está presente em praticamente todas as redes de comunicação de dados, ou seja, os sistemas de voz sobre IP podem ser implementados tanto em linhas de dados privadas, por empresas que interligam suas filiais a matrizes (**Figura 1**), quanto em redes públicas, como a Internet.



Figura 1. VoIP sobre linha de dados privada

Mas, como aproveitar um telefone comum em um sistema de voz sobre IP? Existem diversas maneiras: a solução mais simples é usar um dispositivo chamado ATA (*Analog Telephone Adapter* – adaptador de telefone analógico). Para as empresas que possuem o PABX (*Private Automatic Branch eXchange*) convencional, existem diversos módulos que permitem a conversão da voz para o sinal digital, permitindo a utilização das tecnologias VoIP.

Já as soluções de telefonia IP substituem o PABX convencional por novos equipamentos, totalmente baseados em voz sobre IP, que implementam serviços semelhantes como: transferência entre ramais, conferência de usuários, desvios (siga-me), espera de chamadas, entre outros. Os telefones também são substituídos por equipamentos digitais com comunicação IP nativa. A **Figura 2** ilustra as diferenças entre VoIP e telefonia IP: observe que esta última também utiliza a voz sobre IP.

O PABX IP, como é chamado este novo equipamento, utiliza o hardware de um servidor padrão (como aquele empregado para disponibilizar um site web ou uma plataforma de e-mail) no qual

é instalado o software que implementa as funcionalidades de uma central telefônica. Neste ponto, estão algumas das vantagens da telefonia IP, pois, por se tratar de um servidor comum, o hardware possui custo reduzido.

A expansão do número de usuários (ramais) também se torna uma tarefa simples, dependendo apenas da atualização de recursos como memória RAM (*Random-Access Memory*) e processador. Além disso, é possível aumentar gradativamente o número de servidores que desempenham esta função. Quando utilizado um PABX convencional, a expansão fica restrita ao número máximo de módulos suportado pelo gabinete do equipamento, que é compartilhado por placas para ramais e troncos (**Figura 3**).

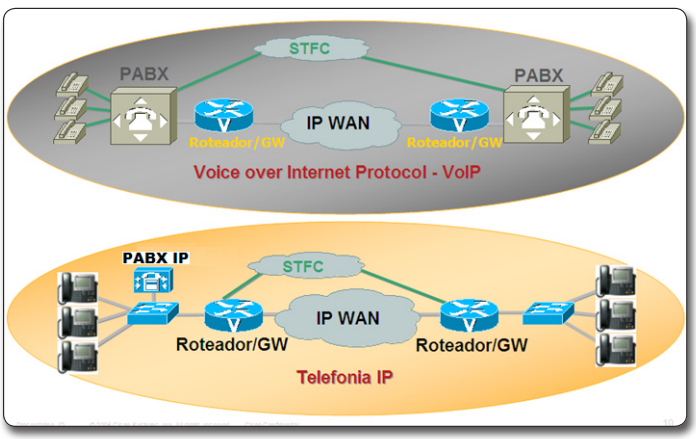


Figura 2. Comparação entre VoIP e telefonia IP

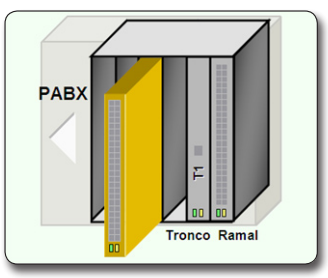


Figura 3. Mix de placas para ramais e troncos

A telefonia IP também é especialmente interessante para empresas com perspectiva de expansão de seus escritórios ou filiais. Ao empregar esta tecnologia, não é necessário o investimento em equipamentos PABX, como ocorre quando utilizada a tecnologia convencional, desde que haja comunicação de dados entre todos os escritórios. Assim, é possível aproveitar o PABX IP já existente na matriz da empresa e realizar apenas o investimento nos telefones IP. Estes podem compartilhar os mesmos recursos implementados para os demais telefones IP já instalados. A solução é totalmente transparente para os usuários (**Figura 4**) e a expansão pode ser realizada com investimentos reduzidos.

A operação de um PABX IP também é bastante simplificada, pois diversas soluções permitem o acesso remoto através de interfaces web amigáveis e intuitivas. É possível realizar o gerenciamento, a configuração e a manutenção a partir de qualquer localidade da empresa. Caso a empresa disponha do recurso de VPN (*Virtual Private Network* – rede privada virtual), o acesso pode ser realizado a partir de qualquer local com conectividade à Internet. Deste modo, são reduzidos os custos de viagem e deslocamento, e as ativações de novas filiais também são realizadas muito mais rapidamente.

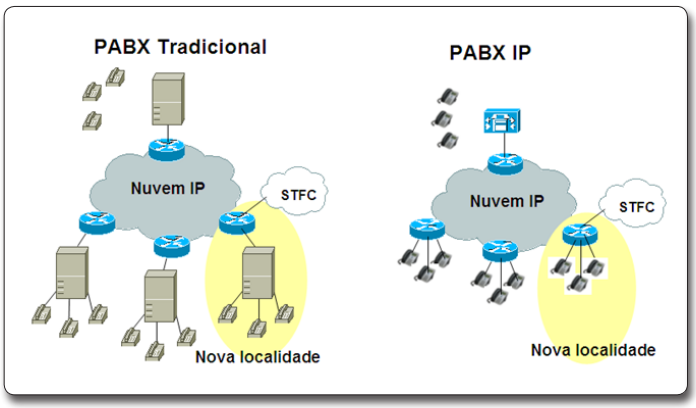


Figura 4. Expansão das filiais utilizando o PABX tradicional e o PABX IP

Diversos modelos de telefones IP são clientes DHCP (*Dynamic Host Configuration Protocol*), ou seja, estes podem ser configurados automaticamente, sem a necessidade de intervenção manual. Desta maneira, para ativar uma nova filial, basta enviar os equipamentos para a localidade e solicitar sua instalação ao suporte local. A maior parte dos modelos permite o uso de um único ponto de rede: este é conectado ao telefone IP e deste, existe uma ligação à estação do usuário. Assim, não são necessários investimentos para expandir o número de pontos de rede. Alguns equipamentos sequer necessitam de fonte de alimentação, porque suportam PoE (*Power over Ethernet* – alimentação elétrica enviada pelo switch através da conexão de rede). A partir daí, o administrador do PABX IP pode realizar a configuração e os testes remotamente.

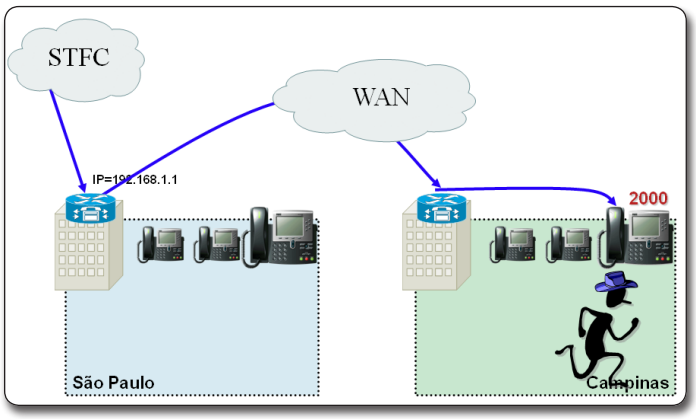


Figura 5. Mobilidade dos usuários

Outro recurso atraente oferecido pelas soluções de telefonia IP é a mobilidade. Os usuários podem utilizar seu ramal para receber ou realizar ligações a partir de qualquer localidade da empresa.

Quando empregada a telefonia convencional, existe a necessidade de conexão física entre o telefone e o PABX, limitando a mobilidade do usuário. Esta restrição deixa de existir na telefonia IP, pois basta conectividade através da rede de dados com o PABX IP.

De tal modo, suponha um profissional de uma empresa contratado para trabalhar na matriz em São Paulo; este trabalha na área comercial e divulga para os clientes o número (11) 7777-2000. Semanalmente, este profissional realiza reuniões com toda a equipe na filial em Campinas. Com a mobilidade, este profissional pode ter seu ramal ativo em qualquer telefone IP instalado na filial. As ligações recebidas através do STFC (*Serviço Telefônico Fixo Comutado*) em São Paulo serão encaminhadas através da rede de dados IP até Campinas (**Figura 5**). Tanto para o profissional quanto para seus clientes, não existem diferenças devido à localização geográfica. Cabe mencionar também que já existem no mercado diversos telefones IP wireless, compatíveis com o padrão WIFI (*Wireless Fidelity*) que possibilitam aos usuários se deslocarem por qualquer local da empresa que possua cobertura da rede de dados sem fio.

A mobilidade também pode ser empregada quando a empresa oferece o recurso de VPN para seus profissionais. Através deste, é possível a partir de qualquer local com conectividade à Internet utilizar um telefone IP (hardphone). É importante destacar que estes usuários remotos ainda podem usar soluções de softphones (softwares aplicativos que emulam as mesmas funcionalidades de um telefone IP).

Por fim, a telefonia IP possibilita a integração com meios de comunicação como correio de voz, e-mail e fax em um único repositório (**Figura 6**). É possível receber as mensagens do correio de voz através de e-mails, acessíveis a partir de qualquer computador conectado à Internet. Por outro lado, os e-mails também podem ser lidos através de qualquer telefone (inclusive externamente à empresa). Os faxes podem ser enviados por intermédio de um simples e-mail com os arquivos anexados, bem como os ramais dos usuários podem ser divulgados como receptores de faxes, que são encaminhados automaticamente por e-mail ao destinatário.

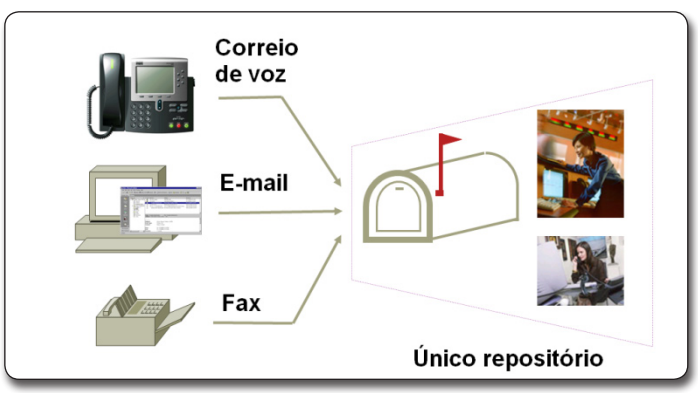


Figura 6. Integração da telefonia IP com correio de voz, e-mail e fax

São inúmeras as vantagens da telefonia IP em relação aos sistemas convencionais de telefonia. A implementação desta solução pode seguir filosofias distintas: processamento centralizado, centralizado e sobrevivência, descentralizado, entre outras. Cada modelo oferece diferentes níveis de segurança e redundância (resiliência), conforme será exposto na sessão a seguir.

Modelos de implementação

Existem diversos modelos que podem ser utilizados para a implementação de uma solução de telefonia IP. Estes devem ser analisados conforme o nível de resiliência desejado (redundância e segurança):

• **Processamento centralizado:** Neste modelo, o PABX IP é instalado em um ponto central, geralmente a matriz da empresa, onde está localizado o datacenter e existe boa infraestrutura para a rede de dados, elétrica e de ar-condicionado. Usualmente todos os enlaces de dados com as filiais também estão concentrados neste local. Conforme apresentado anteriormente, nesta implementação, a administração é centralizada e possibilita a rápida expansão de novas filiais, pois somente será necessária a aquisição de novos telefones IP (**Figura 7**). A mobilidade estará disponível para os profissionais da empresa em todas as localidades. É interessante observar que neste tipo de implementação também é possível criar um plano de numeração unificado que permite realizar uma chamada para qualquer filial discando somente o ramal do usuário;

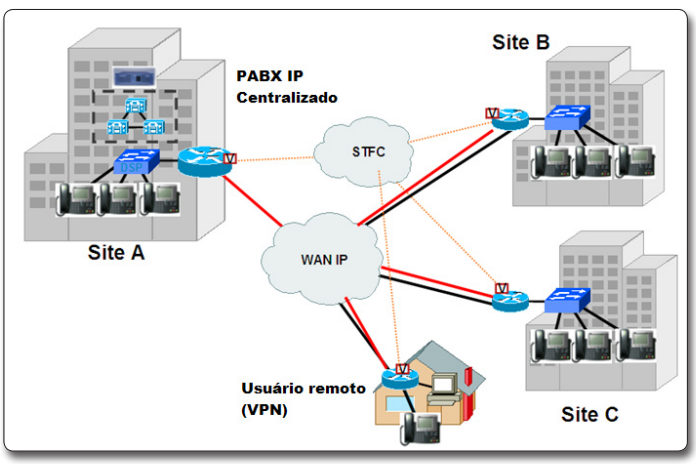


Figura 7. PABX IP instalado na matriz da empresa (ponto central)

• **Processamento centralizado e sobrevivência:** Este modelo possui as mesmas características do modelo anterior, com a exceção da instalação de equipamentos de sobrevivência nas filiais. Nas soluções de alguns fabricantes como Avaya e Cisco Systems, esta funcionalidade geralmente é implementada pelo hardware (também chamado de gateway) no qual são conectados os canais de voz (E1s). Caso ocorra a indisponibilidade na comunicação de dados com as filiais, este entra em modo de contingência automaticamente. Neste modo, somente as funções básicas do PABX IP são oferecidas;

• **Processamento descentralizado:** Neste modelo, existe a redundância geográfica para os equipamentos que compõem a solução de telefonia IP. Tantos os servidores com a função de PABX IP, quanto a conexão com STFC estão distribuídos entre duas ou

mais localidades, formando um cluster descentralizado (**Figura 8**). Caso ocorra a indisponibilidade na comunicação de dados entre as localidades, o PABX IP local manterá operacionais todos os recursos. Neste modelo descentralizado, a administração permanece centralizada (alterações podem ser realizadas a partir de uma interface web única para qualquer localidade), a expansão continua simplificada (não é necessário instalar um PABX IP em todas as localidades) e os usuários podem se locomover por qualquer uma das filiais mantendo seus ramais (mobilidade), além do plano de numeração unificado.

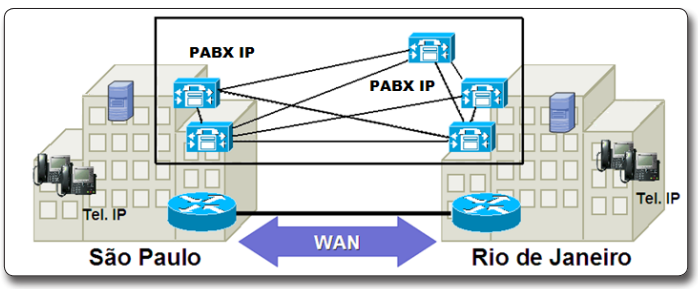


Figura 8. PABX IP instalado em duas localidades (redundância geográfica)

Independentemente do modelo adotado, as soluções de telefonia IP têm grande dependência das redes de dados, pois os sinais analógicos de voz são convertidos para sinais digitais, os quais são transmitidos através desta infraestrutura. As características nativas destas redes impõem grandes desafios às tecnologias que utilizam voz sobre IP, pois estas apresentam requisitos quanto às perdas de pacotes, atrasos e jitter, conforme será apresentado na sessão a seguir.

Desafios das tecnologias que utilizam voz sobre IP

A conversão do sinal analógico de voz para digital possibilitou sua transmissão através das redes de comunicação de dados. A primeira codificação digital de voz data de 1928, na área militar, mas somente na década de 1970 houve aplicação em outras áreas. A técnica inicialmente empregada foi a PCM (*Pulse Code Modulation*), que implementava os seguintes passos básicos:

1. Amostragem do sinal analógico: Nesta etapa, são retiradas amostras periódicas do sinal de voz, que o discretizam no tempo;

2. Quantificação das amostras: Após a amostragem do sinal, é necessário transformar os valores infinitos das amostras em finitos, para que estes possam ser representados pelo sistema binário;

3. Codificação em valores binários: Os valores transformados precisam ser codificados em sequências de bits, pois um sinal digital binário somente pode assumir dois valores diferentes: 0 ou 1.

As técnicas para converter os sinais de áudio para a forma digital evoluíram e passaram a realizar a compressão das amostras para redução da largura de banda necessária para transmissão. Estas também são conhecidas como codecs (codificador-decodificador), destacando-se os seguintes:

• **G.711:** Lançado em 1972, consiste em 8000 amostragens do sinal de voz por segundo, representadas por um valor discreto em 8 bits, resultando na necessidade de um canal digital de 64 kbps para transmissão de cada canal de voz. Existem dois algoritmos básicos definidos como padrão: o algoritmo mu-law (mais comum na América do Norte e no Japão) e o algoritmo a-law (utilizado na Europa e no resto do mundo);

• **G.729:** Codec mais comum, pois necessita uma pequena largura de banda (8 kbps). Compacta o áudio em partes de 10 milissegundos. Músicas, tons DTMF (*Dual-Tone Multi-Frequency*) e faxes não podem ser transportados confiavelmente por este codec, devendo ser usado o codec G.711 ou outro que utilize mais banda para transportar estes tipos de sinais.

Como cada tipo de codec possui uma exigência de largura de banda, a qualidade do sinal de voz resultante apresenta diferentes níveis de qualidade. O método de classificação mais comum emprega uma pontuação entre 1 e 5, onde um grupo de pessoas escuta amostras de voz decodificadas (sendo as mais variadas e envolvendo proporcionalmente vozes masculinas e femininas).

A média das pontuações gera um valor denominado MOS (*Mean Opinion Score*), cujo valor médio é 4,3 para o codec G.711 e 3,92 para o G.729. O valor do MOS entre 4,4 e 4,5 é considerado equivalente à qualidade obtida em uma chamada na STFC, já o MOS em 4,0 ainda é considerado aceitável para a maioria dos usuários. A maior parte das chamadas de dispositivos móveis (celulares) possui MOS entre 3,8 e 4,0 (a voz do originador e o reconhecimento de palavras já podem ser afetados).

Desta maneira, um fator relevante para a implementação das redes de voz sobre IP é o planejamento de sua capacidade, visto que cada codec requer uma determinada largura de banda para cada chamada que será estabelecida (**Tabela 1**).

Codec	G.711	G.726 r32	G.726 r24	G.726 r16	G.728	G.729	G.723 r63	G.723 r53
Bandwidth	64 kbps	32 kbps	24 kbps	16 kbps	16 kbps	8 kbps	6.3 kbps	5.3 kbps

Tabela 1. Largura de banda necessária para diferentes codecs

Deste modo, o uso das tecnologias de voz sobre IP tem como grande desafio as limitações das redes de dados. O protocolo IP é conhecido como best effort (melhor esforço), ou seja, as transmissões dos diversos usuários da rede são concorrentes entre si, resultando no compartilhamento da largura de banda disponível entre todos os fluxos de dados; caso ocorram congestionamentos na rede, os pacotes excedentes são descartados.

Este comportamento é indesejável para aplicações em tempo real (real time), como a voz sobre IP, porque estas requerem uma largura de banda mínima e constante. O protocolo IP também não é orientado à conexão, logo não existe a definição de um caminho único entre emissor e receptor; os pacotes de dados podem passar por enlaces de dados e roteadores com capacidades totais díspares e, consequentemente, níveis de congestionamento e processamento diferentes (**Figura 9**).

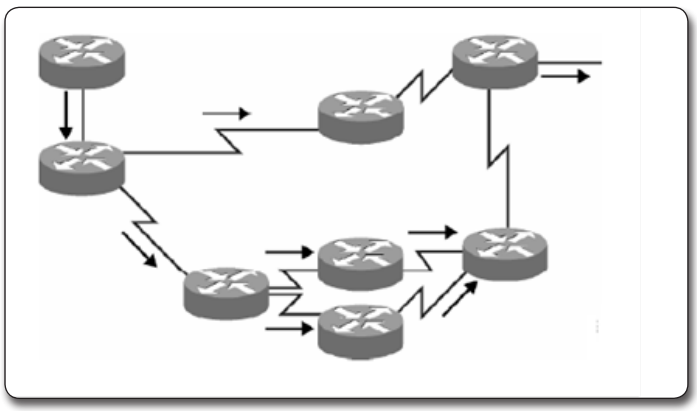


Figura 9. Múltiplos caminhos entre emissor e receptor

Neste contexto, as redes de dados IP devem ser capazes de atender aos seguintes requisitos das tecnologias de voz sobe IP:

• **Perda de pacotes:** Ocorre quando a qualidade da rede é ruim ou se existe congestionamento em algum ponto desta, pois nestes casos os pacotes excedentes são descartados; degrada a qualidade da voz e provoca perdas no áudio transmitido – cortes no sinal de voz (**Figura 10**);

"Bo- dia, c--o va- v--cê?"

Figura 10. Representação gráfica da perda de pacotes

• **Atraso:** Também chamado de atraso fim-a-fim, consiste em dois componentes principais: atraso fixo de rede (processamento dos pacotes nos roteadores e propagação do sinal) e atraso variável de rede (congestionamentos e serialização dos pacotes de dados). As aplicações de voz toleram, tipicamente, até 150 milissegundos de atraso sem que a qualidade da chamada seja degradada (sem percepção dos usuários). O efeito é semelhante ao observado em ligações internacionais, quando um usuário aguarda alguns instantes até que sua mensagem chegue ao destinatário e este envie sua resposta (**Figura 11**);

"..... Bom dia, como vai você?"

Figura 11. Representação gráfica do atraso

• **Jitter:** Corresponde à variação entre a chegada prevista para um pacote IP e o tempo em que este efetivamente foi recebido (variação no atraso devido aos congestionamentos). Para os usuários, o efeito é semelhante ao observado no item anterior (**Figura 12**).

"Bom..... dia, como .. vai ..... você?"

Figura 12. Representação gráfica do jitter

Para tanto, alguns mecanismos foram criados com o objetivo de melhorar as características nativas do protocolo IP em relação aos requisitos das tecnologias IP. O protocolo RTP (*Real-time Transport Protocol*) foi projetado para permitir a compensação do jitter e da desordenação de pacotes introduzida pelos múltiplos caminhos nas redes IP; contudo não garante a entrega dos pacotes nem a qualidade de serviço (QoS).

O RTP é executado sobre o UDP (*User Datagram Protocol*), um protocolo rápido, simples, sem conexão e que provê o transporte fim-a-fim dos dados multimídia, fazendo uma interface entre a camada de aplicação e de transporte.

Os cabeçalhos dos protocolos de transporte não fornecem nenhuma informação sobre o tempo de transmissão dos pacotes. O RTP possui em seu cabeçalho campos específicos para as informações de tempo (timestamp) e numeração de sequência, que são usadas para determinar o jitter e problemas com a ordenação dos pacotes. Outros serviços implementados são: identificação do tipo de payload (dados transportados) e entrega monitorada.

Outro mecanismo criado é a compressão do cabeçalho, pois a soma dos cabeçalhos dos protocolos IP (20 bytes), UDP (8 bytes) e RTP (12 bytes) resulta em 40 bytes. Pode-se observar que a relação entre a quantidade de bytes de dados (parte útil) e de controle (cabeçalhos) é muito alta, pois, por exemplo, o codec G.729 gera somente 20 bytes de dados para serem transmitidos. Assim, uma técnica adotada é a compressão dos cabeçalhos para 2 ou 4 bytes, dependendo do uso de checksum[1] pelo UDP (**Figura 13**). A ideia básica é que após a transmissão do primeiro pacote descomprimido, vários campos dos pacotes seguintes possam ser suprimidos, sendo remontados no destino.

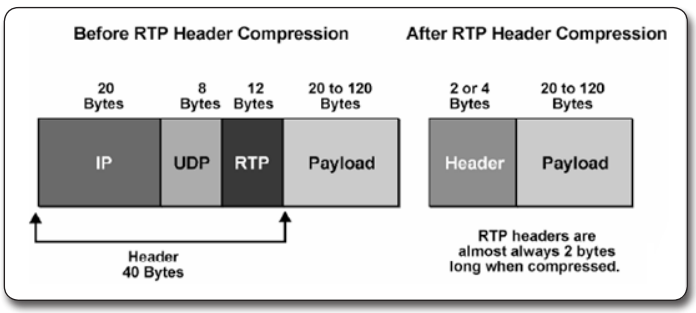


Figura 13. Compressão do cabeçalho RTP

São grandes os desafios das tecnologias que utilizam voz sobre IP. Entretanto, é possível observar sua importância à medida que diversas empresas sólidas na área de telecomunicações já oferecem soluções de telefonia IP, entre as quais, Cisco, Avaya, Nortel e Siemens. A sessão a seguir descreverá as principais características e custos destes produtos.

Algumas soluções de telefonia IP

A seguir são apresentadas algumas soluções de telefonia IP atualmente disponíveis no mercado (segundo o site especializado VoIP-News):

1 Checksum ou soma de verificação é um código usado para verificar a integridade dos dados transmitidos. No protocolo UDP é calculada utilizando seu cabeçalho e a área de dados.

• Fornecedor: Cisco.
– Produto: Unified Communications Manager 6.0 com servidor da família MCS 7800;
– Clientes: Médias e grandes empresas;
– Plataforma: Baseada em Intel;
– DAC (Distribuidor Automático de Chamadas): Sim – opcional;
– URA (Unidade de Resposta Audível): Sim – opcional;
– Redundância de hardware: Sim;
– Suporte a aplicações de terceiros: Sim;
– Instalação: Sistema complexo;
– Custo: A partir de US$ 400,00 por usuário (dependendo da solução escolhida).

•Fornecedor: Avaya.
– Produto: Communication Manager 5.0 com servidor S8730 e Media Gateway G650;
– Clientes: Médias e grandes empresas;
– Plataforma: Servidor Intel Xeon;
– DAC (Distribuidor Automático de Chamadas): Sim;
– URA (Unidade de Resposta Audível): Sim – opcional;
– Redundância de hardware: Sim;
– Suporte a aplicações de terceiros: Sim, através do Application Enablement Services ou do Avaya DeveloperConnection Program;
– Instalação: Simplificada;
– Custo: A partir de US$ 15.000,00.

•Fornecedor: Nortel.
– Produto: Communication Server 1000 Release 5.0;
– Clientes: Médias e grandes empresas;
– Plataforma: Proprietária;
– DAC (Distribuidor Automático de Chamadas): Sim;
– URA (Unidade de Resposta Audível): Sim;
– Redundância de hardware: Sim;
– Suporte a aplicações de terceiros: Sim, através Nortel Partner Program;
– Instalação: Oferece interfaces web e diversos assistentes de configuração;
– Custo: A partir de US$ 250,00 por usuário (dependendo da solução escolhida).

•Fornecedor: Siemens.
– Produto: HiPath 8000;
– Clientes: Médias e grandes empresas;
– Plataforma: Servidor IBM System x3650 T;
– DAC (Distribuidor Automático de Chamadas): Sim – opcional;
– URA (Unidade de Resposta Audível): Sim – opcional;
– Redundância de hardware: Sim;
– Suporte a aplicações de terceiros: Sim, baseada em SIP;
– Instalação: Pré-instalação com DVD e assistentes de configuração;
– Custo: A partir de US$ 42.250,00.

Conclusões

A telefonia IP é uma tecnologia que utiliza voz sobre IP. Nesta, todos os componentes da infraestrutura, como o PABX IP, os telefones IP, os gateways que recebem os canais de voz (E1s), entre outros, suportam nativamente o protocolo IP. Portanto, a telefonia IP difere-se das soluções comumente conhecidas como VoIP pelo fato de ser mais abrangente: as soluções de voz sobre IP continuam a empregar o PABX e os telefones convencionais (geralmente no ambiente corporativo) ou restringem-se à comunicação ponto-a-ponto entre usuários (Skype, MSN, UOL VoIP, entre outros) ou se limitam à integração com a rede pública de telefonia – STFC – possibilitando que as chamadas originadas em softphones, telefones IP e dispositivos ATA sejam encaminhadas para telefones convencionais (Skype, UOL VoIP etc.).

São inúmeras as vantagens da telefonia IP, entre elas: investimento reduzido no hardware do PABX IP (a maior parte das soluções usa servidores comuns), escalabilidade (expansão de recursos como memória, processador ou inclusão de novos servidores), expansão de filiais (somente é necessário adquirir novos telefones IP), administração centralizada, mobilidade dos usuários, utilização de softphones e hardphones, e integração com sistemas de correio de voz, e-mail e fax. Diversas empresas sólidas na área de telecomunicações já oferecem soluções de telefonia IP, entre as quais, Cisco, Avaya, Nortel e Siemens.

O grande desafio do uso destas tecnologias são as limitações das redes de dados. O protocolo IP é conhecido como best effort (melhor esforço), ou seja, as transmissões dos usuários são concorrentes entre si, resultando no compartilhamento da largura de banda disponível, e consequentemente, influenciando na qualidade da voz transmitida. Entretanto, diversos mecanismos já foram implementados e mostraram-se eficazes para contornar as limitações do protocolo IP. A maior prova disto está no sucesso e na grande aplicabilidade observada nos dias atuais.



André Koide da Silva
andre@aksilva.com
Mestre em Engenharia pela Escola Politécnica da Universidade de São Paulo, especialista em Redes de Computadores pela Universidade Estadual de Campinas e bacharel em Sistemas de Informação pela Universidade Presbiteriana Mackenzie. Atualmente é especialista em redes de computadores pelo Universo Online (UOL), com ampla experiência em projetos e implementação de soluções Cisco Systems, Foundry Networks e Juniper Networks. Atua também como professor na Faculdade Impacta Tecnologia. Trabalhou em empresas como Goldnet TI, EDS (Electronic Data Systems do Brasil), Alog Data Centers e Gennari & Peartree Projetos e Sistemas. Experiência de dez anos na área de redes de computadores e tecnologia da informação, além de quatro anos como docente do ensino superior na área de tecnologia da informação. Certificado Cisco IP Communications Express Specialist, Cisco Certified Network Associate (CCNA), Extreme Network Associate (ENA) e 3Com Certified Wireless Expert.

Links

VoIP News – Large Enterprise PBX Comparison Guide
http://www.voip-news.com/whitepaper/pdf/VoIP-News_Enterprise_PBX_Comparison_Guide_03_08.pdf

Artigo "Voz sobre IP: Uma visão geral", escrito por Nelson Luiz Leal Fernandes
http://mailhost.ravel.ufrj.br/arquivosPublicacoes/nelson_voip.pdf

Alta disponibilidade com GNU/Linux

Mário de Mello Bittencourt Neto

Resumo DevMan

De que se trata o artigo:

Este artigo trata de como planejar e executar as configurações necessárias para montar uma estrutura de alta disponibilidade para serviços, quer sejam de Internet ou não, utilizando GNU/Linux.

Para que serve:

Serve para garantir que os serviços (banco de dados, servidores web etc.) estejam disponíveis para uso pelos Clientes mesmo quando da ocorrência de problemas graves, tais como instabilidade ou falhas de hardware.

Em que situação o tema é úti:

Quando você for responsável por manter um serviço que não pode parar (24 h e 7 d) e/ou o mesmo tiver um alto número de acessos e você desejar minimizar os impactos que uma manutenção ou problema de hardware venha causar.

No ambiente atual, a demanda para que os serviços estejam disponíveis 24x7 é maior e cada vez mais presente em todos os segmentos, da indústria ao varejo. Se antes era um requisito existente ou recurso disponível apenas para grandes ambientes, por conta de seu custo, hoje já é possível torná-lo acessível para uma audiência muito maior através de ferramentas open source.

Neste contexto, este artigo apresentará como implementar um ambiente de alta disponibilidade utilizando apenas soluções open source. Contudo, antes de apresentarmos as ferramentas que estão à nossa disposição, é preciso introduzir alguns termos comuns na área e esclarecer os limites da solução proposta.

Alta disponibilidade e escalabilidade

A alta disponibilidade e escalabilidade são requisitos desejáveis em qualquer sistema, e apesar de serem vistas com frequência juntas, as soluções que visam prover alta disponibilidade não implicam em escalabilidade e vice-versa. Para fins deste artigo, podemos definir que:

- **Alta disponibilidade** é a capacidade de um sistema/servidor estar disponível por grandes períodos de tempo;
- **Escalabilidade** é a capacidade de oferecer serviços com qualidade aceitável mesmo com uma demanda crescente.

Este artigo abordará apenas o primeiro item, uma vez que o segundo requer um conhecimento muito mais específico da aplicação que se deseja atender.

Paralisações

Uma vez que um serviço é colocado à disposição dos usuários, uma pressão, antes inexistente, se manifesta: o serviço deve estar disponível sempre que necessário!

Infelizmente, em qualquer ambiente temos que conviver com eventos que atuam para impedir o objetivo acima de ser satisfeito. Estes eventos são as paralisações dos servidores/serviços.

Para minimizarmos seus impactos temos que entender os tipos de paralisações. Estas podem ser divididas em duas categorias:

- **Não planejadas:** são o resultado de falhas aleatórias (imprevisíveis) dos sistemas, associadas a componentes de hardware ou software. Como exemplo, temos discos rígidos com defeitos físicos, memórias com erros, fontes de alimentação queimadas, entre outros;
- **Planejadas (ou manutenções):** associadas a paralisações agendadas para promover a atualização de hardware ou software. Costumam oferecer menor impacto ao negócio por normalmente serem realizadas em momentos de menor tráfego.

No caso das paralisações não planejadas, você pode ter uma postura preventiva, trocando, por exemplo, componentes antes do tempo médio do aparecimento de problemas. Entretanto, tal atitude não é uma garantia de que o componente não apresentará problemas antes dessa data.



Mas qual o impacto de uma paralisação? Apesar de não ser fácil quantificar o prejuízo de uma paralisação, é possível realizar um simples exercício para se entender o potencial que um problema pode alcançar.

Suponha que você tenha um servidor que demora 5 minutos para completar o processo de reinicialização. Se ele é responsável por atender a 100 requisições por segundo, isso significa que teremos **5 minutos x 60 segundos x 100 requisições por segundo = 30.000 requisições não atendidas!**

Níveis de disponibilidade

Apesar do desejo de termos uma disponibilidade de 100%, este valor não é realizável na prática. Por conta disso, precisamos definir qual o nível de disponibilidade, conhecido como SLA (*Service Level Agreement*[1] – que define os níveis aceitáveis para um serviço), antes de desenharmos a nossa solução.

Usualmente iniciamos estimando o tempo máximo desejável de parada (*downtime*) durante um ano de operação medido em minutos, e encontramos qual a classe de disponibilidade que satisfaz o requisito utilizando a seguinte fórmula:

D = (1 - R) x 365 x 24 x 60

Onde **D** é o tempo máximo de parada em minutos e **R** é a razão entre o tempo disponível e o indisponível. No mercado é comum encontrarmos os níveis conforme a **Tabela 1**.

Disponibilidade (%)	Tempo de parada no ano
99	3,65 dias
99,9	8,76 horas
99,99	52,6 minutos
99,999	5,26 minutos
99,9999	30 segundos

Tabela 1. Níveis de disponibilidade

Como atingir a alta disponibilidade?

Uma das maneiras de se atingir a alta disponibilidade está em se desenvolver ou utilizar sistemas tolerantes a falhas. Nestes sistemas, a falha de um componente não afeta a operação, pois é compensada por outro de maneira transparente e automática.

Se considerarmos apenas o contexto de hardware, os sistemas tolerantes a falhas, normalmente proprietários, possuem partes redundantes, ou seja, componentes adicionais que são acionados quando o principal apresenta falha. Nesta categoria encontramos CPUs, memória, disco, fontes de alimentação e refrigeração.

Por sua característica proprietária, os sistemas tolerantes a falhas possuem custos elevados, o que impede a sua adoção fora de ambientes nos quais recursos de pessoal técnico ou financeiros sejam abundantes. Ou pelo menos impedia até o lançamento de projetos como o Linux HA[2] (*Linux High Availability* – é um conjunto de ferramentas disponíveis para ambiente GNU/Linux).

Com o Linux HA temos acesso a soluções, via software, para alcançar os níveis de disponibilidade sem trazer os altos custos de soluções proprietárias.

1 Um SLA é um acordo negociado entre duas partes, onde uma é o Cliente e outra é o fornecedor de um serviço. Ele registra os parâmetros mínimos de qualidade sobre serviços, prioridades, responsabilidades e garantias.

2 O Linux HA foi criado em 1998 e reúne vários subprojetos de software que visam trazer a alta disponibilidade ao ambiente GNU/Linux. Ele tem como principal software o pacote Heartbeat.

Construindo a solução de disponibilidade

Para a construção de nossa solução nos basearemos na redundância de componentes comuns (*commodity hardware* – hardware comum encontrado no mercado) e de software para detectar a falha e promover a recuperação automática.

Neste artigo assumiremos um ambiente exemplo como visto na **Figura 1**. Em nosso ambiente temos um servidor WWW, que serve os arquivos e aplicativos de nossa empresa para o mundo.



Figura 1. Servidor WWW

Obviamente, como temos apenas um servidor, qualquer mau funcionamento nele que acarrete uma paralisação, programada ou não, irá impedir que os usuários acessem os aplicativos lá hospedados.

A primeira abordagem que podemos assumir é duplicar o hardware e de alguma maneira sincronizar o conteúdo de ambos os servidores. Na **Figura 2** podemos ver tal abordagem utilizando uma ferramenta comum no ambiente Unix, o *rsync*.

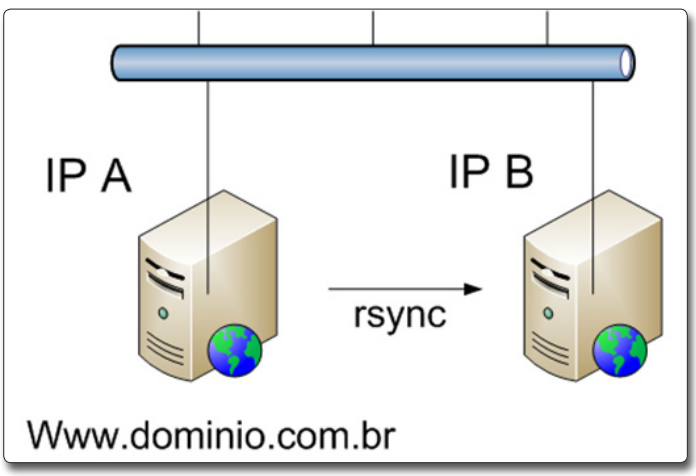


Figura 2. Redundância com o rsync

O *rsync* é uma ferramenta que permite sincronizar o conteúdo de disco (partições, arquivos, etc.) remotamente e de maneira eficiente. Ele só leva as mudanças que aconteceram entre uma sincronização e outra. Assim, se você faz uma sincronização a cada cinco minutos, ele só leva as mudanças ocorridas nesse período, em vez de copiar todo o conteúdo novamente.

Dessa forma, se seu site está configurado para apontar para o IP A e o servidor apresenta problema, você pode reconfigurar a máquina reserva para usar o mesmo IP e assim seus usuários

voltarão a acessar o conteúdo – pelo menos o que existia até a última sincronização (ver **Figura 3**).

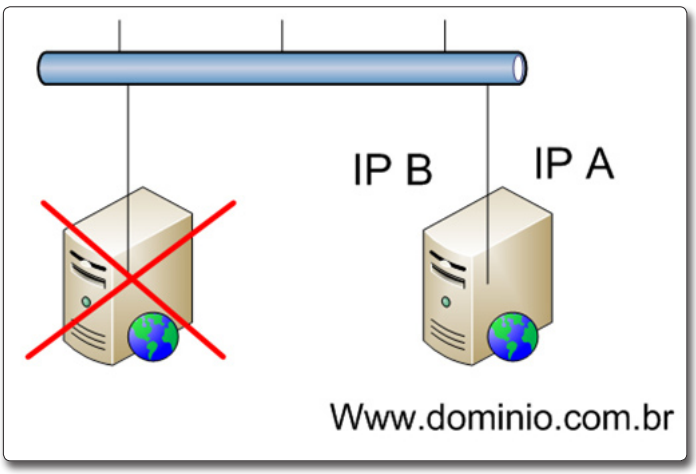


Figura 3. Servidor backup em ação

A solução anterior funciona em alguns ambientes, mas possui muitos limitantes para a realidade pretendida:

• **A sincronização é periódica.** Com isso, todos os dados atualizados entre uma sincronização e outra são perdidos em caso de falha; ou seja, se o servidor de backup sincroniza a cada 1 hora e o principal ficar inacessível nesse intervalo, você colocará o servidor de backup com dados incompletos em seu lugar;

• **A configuração é manual.** A troca do IP do servidor backup e inicialização dos serviços é manual. Isso representa um tempo para recuperação elevado.

Tais restrições nos impendem de alcançar níveis de SLA aceitáveis para a maioria das necessidades. Dessa forma, precisamos utilizar outra abordagem que partirá do mesmo princípio (redundância de hardware), mas se valerá de dois componentes do Linux HA: DRBD (*Distributed Replicated Block Device*) e o Heartbeat para resolver as limitações vistas com a proposta do *rsync*.

O DRBD se apresenta para o sistema operacional como um sistema de blocos, assim como um disco rígido. A diferença é que quando o sistema operacional manda escrever nesse dispositivo ele o faz via rede em um dispositivo remoto. Daí o seu caráter distribuído. Por se apresentar como um sistema de blocos ele é transparente para o sistema operacional, que acredita estar gravando localmente como em qualquer dispositivo.

O Heartbeat, por sua vez, é o nosso agente de monitoramento. Ele fica monitorando os elementos que irão compor nossa solução e ao detectar que o elemento considerado ativo (primário) não está respondendo, ele se encarrega de promover o elemento passivo (secundário) de maneira que o mesmo assuma todos os serviços do elemento ativo.

Nossa meta é criar um ambiente que permita em até 30 segundos detectar a falha e automaticamente possibilitar que o servidor backup assuma os serviços oferecidos pelo servidor principal.

Anatomia da solução

Neste artigo iremos montar um ambiente de alta disponibilidade para um servidor Web. Nossa solução, vista na **Figura 4**, é composta de quatro componentes:

1. Um servidor primário – sistema operacional Linux, com duas interfaces de rede e responsável inicialmente por prover o serviço de Web através do Apache;
2. Um servidor secundário – sistema operacional Linux, com duas interfaces de rede;
3. Um barramento de rede principal ou público – é o barramento de rede pelo qual os Clientes irão acessar o servidor;
4. Um barramento de rede secundário ou privado – é o barramento interno, utilizado apenas para a sincronização (replicação) dos dados do servidor primário para o secundário.

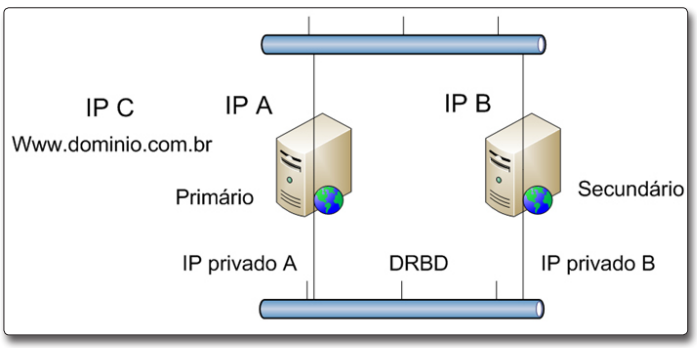


Figura 4. Solução com DRBD+Heartbeat

Na solução usando o DRBD+Hearbeat resolvemos as duas restrições citadas anteriormente (ver **Figuras 2** e **3**):

• **Sincronização dos dados transparente.** Por se comportar como um disco convencional para o sistema operacional (DRBD), toda e qualquer operação de escrita é transmitida para o ponto secundário automaticamente e de maneira síncrona;

• **Configuração automática.** O Hearbeat ao perceber que o ponto primário não está operante, faz com que o nó secundário assuma o seu papel realizando configurações, como de IP, de forma automática.

Na **Figura 5** vemos o que acontece quando ocorre a falha do ponto primário. O servidor B irá assumir o papel do nó primário, passando a responder pelo IP C (associado ao serviço *www.dominio.com.br*) e iniciará o servidor Apache localmente.

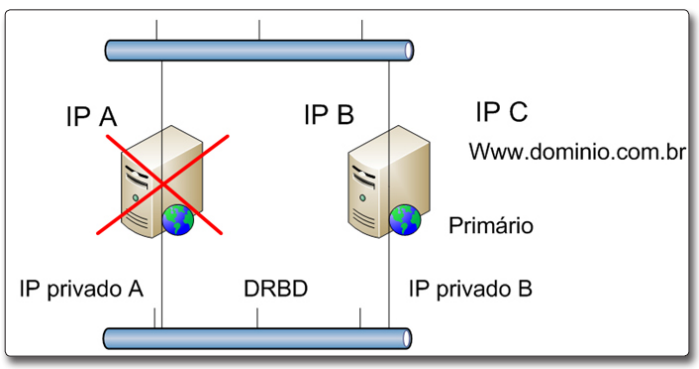


Figura 5. Falha no servidor primário

Configurando o ambiente

Em nosso exemplo estamos montando uma solução HA para um servidor Web. Usaremos nos dois servidores o sistema operacional Linux com a distribuição CentOS 5.x, os pacotes do Apache, DRBD e o Heartbeat.

Para baixar estas aplicações, acesse os endereços listados no final do artigo na seção **Links**.

Apesar dos procedimentos de instalação diferirem quanto à distribuição escolhida, todos os pacotes são encontrados nas distribuições mais recentes.

Para montar um ambiente, usaremos como exemplo os parâmetros abaixo:

Servidor A
Interface pública: 172.16.117.162 / 255.255.255.0
Interface privada: 192.168.0.2 / 255.255.255.0
Nome: node0

Servidor B
Interface pública: 172.16.117.163 / 255.255.255.0
Interface privada: 192.168.0.3 / 255.255.255.0
Nome: node1

IP público do serviço Web
172.16.117.160

É importante que em ambos os servidores o serviço do Apache esteja configurado para não se iniciar automaticamente. Esse controle será feito pelo Heartbeat.

Agora realize as etapas descritas a seguir nos dois servidores, salvo onde explicitamente for indicado para realizar apenas no servidor primário.

1ª Etapa: configurando o DRBD

O DRBD é responsável pela sincronização dos dados entre os nós. É desta maneira que é possível usá-lo sem que as aplicações que fazem uso daquela partição tenham que ser modificadas para se beneficiar da replicação.

O arquivo de configuração do DRBD padrão é o */etc/drbd.conf*. Usualmente é criado um de exemplo no processo de instalação do pacote. Em nosso exemplo teremos o arquivo apresentado na **Listagem 1**.

A explicação sobre o funcionamento de todos os parâmetros está além do escopo deste artigo. No entanto, analisaremos alguns dos parâmetros gerais, isto é, que afetam todos os recursos do DRBD.

Protocol representa o protocolo de controle do status das operações. Ele possui três opções: A, B ou C. O *status* indica se a operação de escrita pode ser considerada completa em ambos nós. Cada protocolo, dentre as opções, oferece mais ou menos segurança quanto a considerar a operação concluída.

No protocolo A, a replicação acontece em modo assíncrono, ou seja, as operações de escrita são consideradas concluídas quando são realizadas no disco local. A ordem de escrita no nó remoto é colocada em uma fila para transmissão na rede. Se ocorrer um erro neste momento é possível haver perda de dados já que os comandos de escrita não chegarão ao nó remoto.

Listagem 1. Arquivo de configuração do DRBD.

```
resource apache
{
 protocol C;
 startup { wfc-timeout 0; degr-wfc-timeout    120; }
 disk { on-io-error detach; }
 net {  cram-hmac-alg "sha1"; shared-secret "senha"; }
 syncer { rate 10M; }

 on node0 {
  device /dev/drbd0;
  disk /dev/sdb;
  address 192.168.0.2:7788;
  meta-disk internal;
 }

 on node1 {
  device /dev/drbd0;
  disk /dev/sdb;
  address 192.168.0.3:7788;
  meta-disk internal;
 }
}
```

No protocolo B, a replicação acontece em modo semi-síncrono, ou seja, as operações de escrita são consideradas concluídas quando são realizadas no disco local e a ordem de escrita no nó remoto chegam ao destino. Ainda é possível haver perda de dados no caso de uma falha simultânea dos nós antes que o nó remoto tenha escrito no seu disco.

No protocolo C, a replicação acontece em modo síncrono, ou seja, as operações de escrita são consideradas concluídas quando ambos os discos (do nó local e remoto) confirmam a gravação dos dados.

Assim, escolhemos o protocolo **C**, pois é o que dá mais garantia quanto à integridade dos dados.

Startup indica parâmetros utilizados na inicialização do DRBD. Em nosso caso, **wfc-timeout** indica o tempo que um nó deve esperar até que o outro nó sinalize estar operacional antes de inicializar o DRBD. Este parâmetro é usado pelo nó sendo inicializado se ele estivesse ativo anteriormente e após um reboot não conseguisse conectar ao outro nó. O valor 0 indica que o nó deverá aguardar indefinidamente ou até uma intervenção manual antes de inicializar o dispositivo.

O **degr-wfc-timeout** é usado em uma situação diferente. Se um nó A estiver inicializando e antes do boot atual o nó B com o qual ele se comunica já estivesse inoperante, o nó A deveria aguardar pelo tempo indicado antes de iniciar o dispositivo DRBD. Em nosso cenário, caso o nó B estivesse fora antes do nó A realizar um reboot, este deveria esperar 120 segundos antes de inicializar os dispositivos.

Disk indica como tratar os avisos que o dispositivo físico reporta ao DRBD. Em nosso caso, se detectar erros de operação de entrada/saída deverá desconectar o dispositivo físico.

Syncer indica a velocidade disponível para o DRBD utilizar na replicação. A velocidade é um fator influenciado pela capacidade de entrada/saída do dispositivo físico (disco rígido/controladora/barramento de dados) e a interface de rede.

Como regra geral, sugere-se limitar em 30% da capacidade nominal de transferência. Por exemplo: suponha que seu sistema de discos suporte uma vazão de 110 MB/s (MBytes/s) e você use uma interface Gigabit na rede de replicação. A taxa a ser usada seria 110 x 0.3 = 33 M.

Net define o algoritmo usado para a autenticação entre os dois nós e qual a senha que usaremos. É uma medida de segurança para evitar que outros equipamentos possam se conectar e interagir de forma não autorizada.

Enquanto os parâmetros anteriores ditam o comportamento geral do nó, temos que definir como os recursos de disco serão usados pelo sistema. Vamos analisar alguns dos parâmetros para a definição dos recursos que serão oferecidos pelo DRBD.

O primeiro parâmetro a ser visto é o **resource**. Cada **resource** deve ser pensado como um nome associado a uma partição de um disco rígido. Da mesma maneira que você pode criar **alias** (apelido) para discos físicos em sua configuração, um **resource** é um indicador que você usará de agora em diante para todos os comandos de manipulação do DRBD.

Você pode ter vários recursos dentro do DRBD. No sistema operacional (e no Heartbeat) você irá referenciar a eles através de um nome para facilitar a execução de comandos. Em nossa solução, definimos um recurso chamado apache:

```
resource apache
```

Na sequência definimos os nós. Abaixo apresentamos um fragmento do mesmo arquivo de configuração, com foco apenas nos parâmetros que definem um recurso:

```
on node0 {
   device /dev/drbd0;
   disk /dev/sdb;
   address 192.168.0.2:7788;
   meta-disk internal;
 }

on node1 {
   device /dev/drbd0;
   disk /dev/sdb;
   address 192.168.0.3:7788;
   meta-disk internal;
 }
```

Em nosso caso temos no primeiro nó (**node0**) um dispositivo */dev/drbd0* que é mapeado para o disco físico */dev/sdb*, e vemos que ele fica escutando no IP *192.168.0.2* na porta *7788*.

Ao observarmos o segundo nó, vemos que com exceção do IP, que é o IP privado do servidor B, as demais configurações são as mesmas.

Para entendermos o que foi feito é preciso explicar que, em nosso exemplo, existe um segundo disco (*/dev/sdb*) em cada servidor e que criamos em cada nó um dispositivo (*/dev/drbd0*) que gravará e lerá os dados desse disco.

O nosso sistema operacional irá gravar na partição **drbd0** achando que se trata de um disco físico local (como */dev/sda*) e os dados serão replicados para o nó que estiver em modo secundário de forma automática. Assim, se houver uma pane no servidor primário, o segundo terá uma cópia exata dos dados até o momento anterior da pane.

Após criar o arquivo de configuração, copie o mesmo manualmente para o segundo servidor (já que ele é idêntico) e execute como usuário root o comando a seguir na console dos dois servidores:

```
drbdadm create-md apache
```

O *drbdadm* é um utilitário que é instalado com o DRBD e lhe permitirá consultar e alterar o comportamento do nó, se necessário.

Neste ponto, o DRBD irá iniciar o processo de inicialização do dispositivo */dev/drbd0*, conforme a **Listagem 2**.

Listagem 2. Processo de inicialização do dispositivo drbd0.

```
v08 Magic number not found
v07 Magic number not found
About to create a new drbd meta data block on /dev/sdb.
. ==> This might destroy existing data! <==
Do you want to proceed? [need to type 'yes' to confirm] yes
Creating meta data... initialising activity log NOT initialized bitmap (256 KB) New
drbd meta data block sucessfully created.
```

O tempo necessário para esse procedimento dependerá do tamanho do disco (ou partição) usado.

Agora você deve iniciar o serviço do DRBD com o comando:

```
service drbd start
```

Se ocorreu tudo bem, ambos os nós estarão marcados como secundários. Então execute o comando abaixo em ambos os servidores:

```
cat /proc/drbd
```

Ele irá mostrar o status do sistema de controle do DRBD para que você veja como se encontram os dispositivos criados, qual o papel de cada dispositivo (primário/secundário) no servidor onde o comando foi executado e se está sincronizado ou não.

O resultado apresentado na **Listagem 3** indica que o dispositivo do DRBD foi criado e que ambos os nós estão no modo secundário.

Agora você deve escolher um dos dois nós para considerar como primário. Como ambos neste momento não possuem nenhum conteúdo, na prática você pode selecionar qualquer um deles. Para padronizarmos, iremos executar o comando abaixo no **node0**:

```
drbdadm -- --overwrite-data-of-peer primary apache
```

O comando irá instruir o servidor onde ele foi executado; em nosso caso **node0**, que ele deve copiar o conteúdo dos blocos do disco do servidor primário para o secundário.

Como no DRBD nós associamos um nome a cada partição criada, estamos fazendo isso para o resource apache. Caso você tivesse mais de um resource, você deveria fazê-lo para todos os resources existentes.

Agora, se repetirmos o *cat /proc/drbd* a partir do **node0**, devemos ver o resultado exibido na **Listagem 4**.

Listagem 3. O processo de inicialização do drbd0 foi concluído.

```
version: 8.0.4 (api:86/proto:86) SVN Revision: 2947 build by buildsvn
@c5-i386-build, 2007-07-31 19:17:18
 . 0: cs:Connected st:Secondary/Secondary ds:Inconsistent/Inconsistent C r---
 . ns:0 nr:0 dw:0 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0
 . resync: used:0/31 hits:0 misses:0 starving:0 dirty:0 changed:
   0 act_log: used:0/257 hits:0 misses:0 starving:0 dirty:0 changed:0
```

Listagem 4. Processo de sincronização entre o nó primário e secundário.

```
0: cs:SyncTarget st:Primary/Secondary ds:Inconsistent/Inconsistent C r---
 . ns:0 nr:68608 dw:68608 dr:0 al:0 bm:4 lo:0 pe:0 ua:0 ap:0
 . [>.....................] sync'ed:  0.9% (8124/8191)M finish: 0:12:05
   speed: 11,432 (11,432) K/sec resync:
  used:0/31 hits:4283 misses:5 starving:0 dirty:0 changed:5 act_log:
   used:0/257 hits:0 misses:0 starving:0
  dirty:0 changed:0
```

Neste momento ele indica que está realizando o processo de sincronização entre o servidor primário e o secundário. No exemplo da **Listagem 4** o DRBD se encontra em 0.9% da sincronização e estima que necessitará de mais 12 minutos e 5 segundos para concluir. O tempo real dependerá do tamanho e performance dos discos de cada servidor e da rede.

Quando o DRBD concluiu o processo de sincronização você ficou com dois dispositivos, um em cada nó, com o mesmo conteúdo de cada bloco do disco físico. Os blocos estão, entretanto, sem nenhuma formatação específica. Para poder usar o disco para armazenar o conteúdo você precisa formatar a partição.

Deste modo, quando o valor estiver em 100%, execute o seguinte comando no **node0**:

```
mkfs.ext3 /dev/drbd0
```

Feito isso, você já pode montar a partição e começar a usá-la.

```
mkdir /mnt/apache
mount /dev/drbd0 /mnt/apache
```

Com isso já conseguimos resolver metade de nosso objetivo. Tudo que venha a ser escrito em */mnt/apache* será replicado automaticamente no **node1**.

Para darmos continuidade ao processo, iremos desmontar a partição */dev/drbd0* recentemente criada. A função do Heartbeat será gerenciar e cuidar de montar e desmontar as partições automaticamente. Para isso, execute:

```
umount /mnt/apache
```

O próximo passo é configurar o Heartbeat para que monitore o sistema e promova o servidor secundário a primário, quando necessário.

2ª Etapa: Configurando o Heartbeat

O Heartbeat é uma ferramenta de monitoramento bastante versátil. Para a finalidade de nossa solução, estaremos interessados em três arquivos de configuração: */etc/ha.d/authkeys*, */etc/ha.d/ha.cf* e */etc/ha.d/haresources*.

No */etc/ha.d/authkeys* você irá configurar um código e uma senha que deverão ser usados por ambos os nós na comunicação do Heartbeat:

```
auth 1
1 sha1 senha-do-heartbeat
```

O **auth 1** indica o código a ser usado para identificar os servidores que participarão da comunicação, e na linha seguinte qual a senha a ser usada para garantir que apenas os nós participantes possam se identificar.

No */etc/ha.d/ha.cf* você deve configurar os parâmetros que serão usados pelo Heartbeat para monitorar os nós e que critérios usar para tomar a decisão se um nó está operante ou não (**Listagem 5**).

Listagem 5. Configuração do Heartbeat.

```
keepalive 1
deadtime 10
warntime 5
initdead 120
udpport 694
ping 172.16.117.1
bcast eth1
auto_failback off
node    172.16.117.162
node    172.16.117.163
```

Aqui os pontos principais são:

- **deadtime:** número de segundos sem comunicação a partir do qual o nó é considerado "morto";
- **ping:** indica qual o endereço IP que o Heartbeat irá monitorar. Em nosso exemplo estamos monitorando o gateway;
- **bcast:** indica qual a interface de rede no qual os pacotes de controle do Heartbeat serão enviados;
- **auto_failback:** caso um nó seja detectado como morto e depois volte à atividade ele deve (ou não) ser auto promovido a primário. Em nosso caso optamos por avaliar manualmente o evento e decidir qual nó deve ser o primário em um caso desses;
- **node:** indica os IPs da rede pública onde o Heartbeat está instalado.

A parte mais importante para nossa configuração está no último arquivo (*haresources*). É nele que indicaremos o que o Heartbeat deverá fazer quando detectar um problema e precisar elevar um nó de secundário para primário e vice-versa:

```
node0 IPaddr::172.16.117.160/24 drbddisk::apache \
Filesystem::/dev/drbd0::/mnt/apache::ext3::defaults httpd
```

Em nosso exemplo estamos indicando que o nó que estiver como primário deverá configurar o IP *172.16.117.160/255.255.255.0*, montar a partição */dev/drbd0* como */mnt/apache* e iniciar o serviço *httpd*.

No manual do Heartbeat estão descritas todas as opções válidas, mas é importante entender que o processamento desse arquivo acontece na seguinte ordem:

• **Da esquerda para a direita e de cima para baixo.** Quando um nó está sendo promovido a primário ele inicia executando os comandos nessa sequência;

Ou:

• **Da direita para a esquerda e de baixo para cima.** Quando um nó está sendo rebaixado de primário para secundário ele inicia executando os comandos nessa sequência.

É importante que entendamos esta ordem e o porquê dela antes de criarmos nosso *haresources*.

Quando um nó está se ativando e for decidido que ele é primário, ele irá começar criando um alias com o IP indicado. Este IP é o nosso IP público associado ao servidor Web *www.dominio.com.br*. Em seguida ele irá montar a partição DRBD no diretório que nós indicamos e só depois irá inicializar o serviço do Apache. Após a inicialização do Apache, quem tentar acessar o serviço irá visualizar a página normalmente.

O nó secundário ficará monitorando e, através do DRBD, sincronizando as modificações na partição criada.

Agora suponha que aconteça algo com o nó primário que o impeça de se comunicar com o gateway, por exemplo: a porta do switch apresente problema ou o cabo de rede seja desconectado.

O Heartbeat detectará este problema e após 10 segundos avisará ao nó secundário que ele deve ser promovido. Esse tempo de 10 segundos serve para evitar que pequenas interrupções temporárias gerem falsos positivos e provoquem a alteração desnecessária. Para isso o Heartbeat irá, no atual nó primário (**node0**), desligar o serviço do Apache, desmontar a partição e remover o alias do IP.

No **node1** ele fará exatamente o que fez quando o **node0** foi ativado, e desta maneira, em menos de 30 segundos, os usuários que estivessem acessando o site *www.dominio.com.br* seriam atendidos pelo novo servidor sem tomar conhecimento da mudança ou qualquer intervenção manual.

3ª Etapa: Testando

Ao chegarmos nessa etapa já temos todos os componentes responsáveis pela alta disponibilidade configurados. O que precisamos fazer agora é realizar ajustes no Apache para que ele trabalhe corretamente com nossa solução.

Deste modo, antes de efetuarmos os testes, existem dois passos necessários:

• Configurar o Apache para disponibilizar o conteúdo da partição criada e não do lugar padrão. Como o Apache por padrão utiliza o diretório */var/www/html*, é necessário alterá-lo para o */mnt/apache*;

• Iniciar o Heartbeat em ambos servidores.

Primeiro configure o Apache para usar o */mnt/apache* como **DocumentRoot**. Isso pode ser feito editando o arquivo de configuração */etc/httpd/conf/httpd.conf*. Assim, procure as entradas **DocumentRoot** e **Directory** que apontarem para */var/www/html* e mude-as para */mnt/apache*.

Em seguida inicie o Heartbeat em ambos os nós. Até o momento ele estava configurado, mas não estava em uso.

```
/etc/init.d/heartbeat start
```

Com isso, o Heartbeat irá ler os arquivos de configuração que apresentamos anteriormente e executar os comandos que criamos no arquivo *haresources*.

Neste momento, se observarmos no **node0** as interfaces de rede através do comando *ifconfig*, é possível notar que foi criado um alias (**eth0:0**) para a interface com o IP público, como destaca a **Listagem 6**.

Listagem 6. Listagem das interfaces de rede e seus endereços após o Heartbeat iniciar o node0.

```
eth0    Link encap:Ethernet  HWaddr 00:0C:29:DA:14:35
inet addr:172.16.117.162  Bcast:172.16.117.255  Mask:255.255.255.0

eth0:0   Link encap:Ethernet  HWaddr 00:0C:29:DA:14:35
inet addr:172.16.117.160  Bcast:172.16.117.255  Mask:255.255.255.0

eth1    Link encap:Ethernet  HWaddr 00:0C:29:DA:14:3F
inet addr:192.168.0.2  Bcast:192.168.0.255  Mask:255.255.255.0
```

É possível ainda verificar que ele montou a partição que irá conter o resultado, através do comando *df*:

```
Filesystem          1K-blocks    Used Available Use% Mounted on
/dev/drbd0           507696      10549   470936   3% /mnt/apache
```

Em seguida verifique que ele iniciou o serviço do Apache. Use o comando *ps –aux | grep httpd*:

```
4528 Ss /usr/bin/httpd
4537 S  /usr/bin/httpd
```

Se observarmos o log do Heartbeat, podemos ver as operações sendo executadas e ele fazendo com que os recursos (IP, disco e serviço) migrem de um nó para o outro. O log fica localizado em */var/log/messages* (veja a **Listagem 7**).

Na parte em negrito da **Listagem 7** vemos que o Heartbeat começou a executar os comandos que configuramos no arquivo *haresources*. Note que o Heartbeat primeiro cria o alias com IP 172.16.117.160.

Na **Listagem 8** vemos que o Heartbeat montou a partição */dev/drbd0* como */mnt/apache*. Ainda na **Listagem 8**, também notamos que o Heartbeat iniciou o serviço do Apache para receber as requisições de páginas Web.

```
node0 ResourceManager: info: Running /etc/init.d/httpd  start
```

Nesse momento, se você acessar o endereço *http://172.16.117.160*, verá o conteúdo que estiver na partição */mnt/apache*.

Para vermos a troca de identidades, ou seja, quem é o nó primário (ativo) e quem é o nó secundário (passivo) basta desconectar o cabo de rede do servidor **node0**.

Ao fazer isso, o **node1** assumirá o papel como nó primário em poucos segundos. Depois acesse o servidor **node1** e veja o conteúdo do arquivo */var/log/messages*.

O log exibido na **Listagem 9** ilustra o momento em que **node1** detecta que **node0** está inoperante e com isso identifica que precisa se apoderar dos recursos.

Se você repetir os comandos vistos quando iniciamos o Heartbeat, mas agora no **node1**, verá que ele assumiu o alias, montou a partição e iniciou o servidor apache.

Listagem 7. O Heartbeat associa o IP 172.16.117.160 ao nó ativo.

```
node0 heartbeat: info: Local Resource acquisition completed.
node0 harc: info: Running /etc/ha.d/rc.d/ip-request-resp ip-request-resp
node0 ip-request-resp: received ip-request-resp IPaddr::172.16.117.160/24 OK yes
node0 ResourceManager: info: Acquiring resource group: node0
IPaddr::172.16.117.160/24 drbddisk::apache
Filesystem::/dev/drbd0::/mnt/apache::ext3::defaults httpd
node0 IPaddr: INFO:  Resource is stopped
node0 ResourceManager: info: Running /etc/ha.d/resource.d/IPaddr
172.16.117.160/24 start
```

Listagem 8. O Heartbeat monta a partição para uso e inicia o Apache.

```
node0 IPaddr: INFO: Using calculated nic for 172.16.117.160: eth0
node0 IPaddr: INFO: Using calculated netmask for 172.16.117.160: 255.255.255.0
node0 IPaddr: INFO: eval ifconfig eth0:0 172.16.117.160 netmask 255.255.255.0
broadcast 172.16.117.255
node0 IPaddr: INFO:  Success
node0 Filesystem: INFO:  Resource is stopped
node0 ResourceManager: info: Running /etc/ha.d/resource.d/Filesystem /dev/
drbd0 /mnt/apache ext3 defaults start
node0 Filesystem: INFO: Running start for /dev/drbd0 on /mnt/apache
node0 kernel: kjournald starting.  Commit interval 5 seconds
kernel: EXT3 FS on drbd0, internal journal
node0 kernel: EXT3-fs: mounted filesystem with ordered data mode.
node0 Filesystem: INFO:  Success
```

Listagem 9. Log do servidor secundário no momento que o Heartbeat detecta que o outro nó está inoperante.

```
node1 heartbeat: info: Link node0:eth1 dead.
node1 ipfail: info: Status update: Node node0 now has status dead
node1 harc: info: Running /etc/ha.d/rc.d/status status
node1 heartbeat: info: No local resources [/usr/share/heartbeat/ResourceManager
listkeys node1] to acquire.
```

Conclusão

Ao longo deste artigo vimos como é possível configurar dois servidores para prover um serviço de alta disponibilidade, com tempo de parada inferior a 30 segundos, utilizando-se apenas de hardware convencional e de software livre.

Os conceitos vistos aqui podem ser utilizados essencialmente em qualquer servidor/serviço. Servidores de arquivo (como Samba ou NFS) e ainda servidores de banco de dados são candidatos ideais e podem ser configurados para trabalhar nessa estrutura com um mínimo de esforço.

A etapa mais importante é a de definir as ações que o Heartbeat deverá executar ao iniciar um nó primário e ao transferir os recursos para o secundário. Isso lhe permitirá identificar as modificações, se houver, nas configurações dos serviços a serem oferecidos e atingir padrões de disponibilidade antes só alcançáveis com grandes investimentos.

Tudo isso com recursos disponíveis a qualquer um com uma conexão à Internet e um pouco de dedicação.



Mário de Mello Bittencourt Neto
mbneto@gmail.com
Diretor técnico da Argo Internet desde 1996, atuando na área de redes e desenvolvimento de aplicativos Web, eCommerce e segurança. Engenheiro eletrônico e Mestre em informática. Zend Certified Engineer.

Links

Linux-HA
www.linux-ha.org

Planet HA
www.planet-ha.org

MySQL com DRBD
www.mysql.com/drbd

DRBD
www.drbd.org

Administrando sistemas remotamente

Flávio Alexandre dos Reis

Uma das grandes vantagens no uso do GNU/Linux é a facilidade de administrar o sistema remotamente, seja administração feita via linha de comando ou com acesso à interface gráfica. Essa é uma necessidade para qualquer administrador de redes que mantém diversos servidores, em diferentes locais.

Nos dias atuais, poucas empresas hospedam seus *websites* em servidores instalados dentro da própria empresa. Quase sempre os servidores ficam hospedados em *data centers* que oferecem toda a estrutura necessária para que os serviços fiquem em funcionamento de forma confiável, incluindo *links* redundantes, *no-breaks* de grande porte, geradores, refrigeração, dentre outras vantagens.

Dessa forma, apesar do servidor ser de propriedade da empresa, o administrador não tem nenhum tipo de acesso físico a ele, tudo terá que ser feito à distância.

No GNU/Linux, toda a configuração do sistema, instalação de novos programas, dentre algumas funcionalidades, podem ser feitas a partir do modo texto, o que permite configurar o servidor e mantê-lo atualizado remotamente, não dependendo de aplicações específicas de modo gráfico. Toda essa manutenção poderá ser efetuada de outro sistema operacional. Por exemplo, uma estação Windows pode efetuar um acesso remoto a um servidor GNU/Linux, possibilitando assim a manutenção desejada.

Um exemplo comum no uso de acesso remoto é dentro das redes locais para acesso a aplicativos. Em muitas situações, faz sentido instalar determinados aplicativos em um servidor central e abrir seções remotas nos clientes, permitindo assim a centralização de informações no servidor (facilitando os *backups*, por exemplo) e, ao mesmo tempo, usar menos recursos nos clientes, o que possibilita o uso de micros mais antigos.

TELNET

O serviço *telnet* é um protocolo de acesso remoto cliente/servidor que permite ao administrador trabalhar conectado a outro computador à distância como se estivesse em frente ao mesmo. Esse protocolo chegou para substituir o *rlogin* e possui muitas melhorias em relação a seu antecessor, como o controle de acesso, personalização de seção e controle de terminal.

O *telnet* viabiliza emular terminais, permitindo assim ao usuário acessar aplicações remotas que estejam disponibilizadas em outras redes. A partir do momento que se inicia a seção de trabalho, qualquer comando digitado é enviado diretamente para o computador remoto. Será solicitado um *username* e uma *password* para acessar o sistema, conforme visto na **Figura 1**.

Resumo DevMan

De que se trata o artigo:

Este artigo tem como objetivo demonstrar de forma prática uma técnica capaz de confundir scanners mal intencionados na procura por serviços de acesso remoto ativos no servidor, tornando a porta de acesso remoto do SSH (Secure Shell) oculta para estas ameaças. A técnica de port knocking, aliada às configurações de segurança do SSH, cria um canal confiável de comunicação para servidores GNU/Linux.

Para que serve:

A técnica de port knocking está intimamente ligada ao SSH, que simultaneamente é um programa e um protocolo de rede capaz de permitir conexões remotas por meio das redes de computadores, de forma a permitir a execução de comandos em um computador remoto. O SSH vem se tornando a cada dia mais popular entre os administradores de sistemas e, por conta de sua facilidade de uso e por possuir diversos sistemas clientes, proporciona uma grande flexibilidade e segurança na autenticação e proteção do tráfego de dados.

Em que situação o tema é útil:

A técnica port knocking, e procedimentos relacionados, podem ser aplicados em servidores que necessitam de conexão remota para a administração. Implementando mecanismos e configurações de segurança, o usuário administrador não irá efetuar login diretamente no sistema remoto. Pode-se criar uma configuração automática de lista negra de endereços IP, tempo de inatividade de seção, ocultamento da porta de conexão remota, dentre outras técnicas que serão apresentadas posteriormente.

```
flavio@ubuntu:~$ telnet 192.168.105.100
Trying 192.168.105.100...
Connected to 192.168.105.100.
Escape character is '^]'.
Debian GNU/Linux 5.0
debian login:
```

Figura 1. Início da conexão com servidor telnet

O protocolo *telnet* oferece um terminal virtual de rede, que proporciona uma interface padrão para aplicações remotas, não havendo necessidade de um programa cliente compreender os detalhes de todos os possíveis sistemas remotos. Inclui um mecanismo que permite ao cliente e ao servidor negociarem opções e proporcionar um conjunto de opções padrão. Trata ambas as pontas da conexão simetricamente. Assim, ao invés de forçar o

cliente a conectar-se a um terminal de usuário, o protocolo *telnet* permite um programa arbitrário tornar-se um cliente.

A seguir são listadas algumas características do protocolo *telnet*:

- Conexão rápida (não utiliza transmissão de dados criptografada), recomendado para ambientes seguros;
- Possui uma versão com suporte a criptografia via *ssl*;
- Possui controle de acesso *tcpd* (usando */etc/hosts.allow* e */etc/hosts.deny*);
- A maioria dos sistemas operacionais possui esse utilitário por padrão como sistema de acesso remoto a máquinas UNIX;
- Suporte a terminais ANSI (cores e códigos de escape especiais para o console) e uma grande variedade de outros terminais;
- O *telnet* responde por padrão na porta 23/tcp.

Um servidor *telnet* poderá criar arquivos de *log* a fim de auditar o sistema. As mensagens do servidor *telnet* relacionadas com seções são enviadas para */var/log/daemon.log*, conforme visto na **Figura 2**.

As mensagens sobre autenticação e serviços de *login* são registradas pelos módulos PAM (Sistema padrão de autenticação do GNU/Linux) em */var/log/auth.log*, conforme visto na **Figura 3**.

```
debian:~# tail -n 5 /var/log/daemon.log
Jun 27 19:51:30 debian in.telnetd[2252]: connect from 192.168.105.5 (192.168.105.5)
Jun 27 20:01:56 debian in.telnetd[2270]: connect from 192.168.105.5 (192.168.105.5)
Jun 27 20:02:38 debian in.telnetd[2286]: connect from 192.168.105.5 (192.168.105.5)
Jun 27 20:15:28 debian in.telnetd[2304]: connect from 192.168.105.5 (192.168.105.5)
Jun 27 20:17:00 debian in.telnetd[2321]: connect from 192.168.105.5 (192.168.105.5)
debian:~#
```

Figura 2. Exemplo do arquivo /var/log/daemon.log

```
debian:~# tail -n 10 /var/log/auth.log
Jun 27 20:17:01 debian CRON[2323]: pam_unix(cron:session): session opened for us
er root by (uid=0)
Jun 27 20:17:01 debian CRON[2323]: pam_unix(cron:session): session closed for us
er root
Jun 27 20:17:03 debian login[2322]: pam_unix(login:session): session opened for
user flavio by (uid=0)
Jun 27 20:23:29 debian su[2355]: pam_unix(su:auth): authentication failure; logn
ame=flavio uid=1000 euid=0 tty=pts/1 ruser=flavio rhost=  user=root
Jun 27 20:23:32 debian su[2355]: pam_authenticate: Authentication failure
Jun 27 20:23:32 debian su[2355]: FAILED su for root by flavio
Jun 27 20:23:32 debian su[2355]: - pts/1 flavio:root
Jun 27 20:23:35 debian su[2357]: Successful su for root by flavio
Jun 27 20:23:35 debian su[2357]: + pts/1 flavio:root
Jun 27 20:23:35 debian su[2357]: pam_unix(su:session): session opened for user r
oot by flavio(uid=1000)
debian:~#
```

Figura 3. Exemplo do arquivo /var/log/auth.log

O programa *tcpd*, que pode ser configurado no arquivo */etc/inetd.conf*, oferece mecanismos de registro e controle de acesso para os serviços que está configurado para proteger. Ele é um tipo de *firewall* simples, de fácil configuração, que pode evitar conexões indesejadas e ainda registrar possíveis tentativas de invasão. Quando é executado pelo programa *inetd* (veja a **Nota DevMan 1**), lê os arquivos contendo regras de acesso, permitindo ou bloqueando o acesso ao servidor. Se nenhuma regra conferir, então assume que o acesso deve ser permitido a qualquer um. Esse tipo de bloqueio é conhecido como *TCP WRAPPERS* e pode ser configurado nos arquivos */etc/hosts.allow* e */etc/hosts.deny*.

O arquivo *hosts.allow* contém regras descrevendo que *hosts* têm permissão de acessar um serviço na máquina. O formato do arquivo é muito simples, como visto na **Listagem 1**.



Nota do DevMan 1

inetd

É chamado de "Super-Servidor Internet" porque gerencia conexões para diversos daemons. Daemons são processos que rodam em segundo plano esperando por requisições.

Listagem 1. Estrutura do arquivo hosts.allow.

```
# /etc/hosts.allow
#
# lista de serviços: lista de hosts: comando
```

No campo lista de serviços será inserida uma lista com seus nomes (separados por vírgula), aonde essa regra irá se aplicar. Alguns exemplos são: *ftpd*, *telnetd* e *fingerd*.

Na lista de *hosts* do arquivo, pode-se especificar nomes de computadores ou endereços IP usando caracteres coringas para atingir grupos de *hosts*. O arquivo poderá conter um endereço específico ou combinações de *strings* para atingir qualquer endereço de computador. Alguns parâmetros especiais também podem ser utilizados para simplificar a configuração, são eles: *ALL*, atinge todos os endereços; *LOCAL*, atinge qualquer computador que não contém um "." (isto é, está no mesmo domínio de sua máquina); e *PARANOID*, que atinge qualquer computador que o nome não confere com seu endereço (falsificação de nome). Existe também um último parâmetro, o *EXCEPT*, útil quando se deseja criar uma lista de exceções.

A opção "comando" na **Listagem 1** é um parâmetro opcional. Nesse parâmetro é inserido o caminho completo de um comando que deverá ser executado toda vez que a regra for utilizada. Nesta opção pode ser executado um comando para tentar identificar quem está conectado pelo *host* remoto, para gerar uma mensagem via e-mail ou algum outro alerta para um administrador de rede informando que alguém está tentando se conectar.

Existe um grande número de opções que podem ser incluídas. Alguns exemplos comuns são: com a opção "%h" será inserido o nome do computador que está conectado. A opção "%d" insere o nome do *daemon* a ser chamado. Se o computador tiver permissão de acessar um serviço através do */etc/hosts.allow*, então o */etc/hosts.deny* não será consultado e o acesso será permitido. Um exemplo do */etc/hosts.allow* pode ser observado na **Listagem 2**.

Listagem 2. Exemplo do arquivo hosts.allow.

```
# /etc/hosts.allow
#
# Permite que qualquer um envie e-mails
in.smtpd: ALL
# Permite telnet e ftp somente para hosts locais e myhost.athome.org.au
in.telnetd, in.ftpd: LOCAL, myhost.athome.org.au
# Permite finger para qualquer um, mas manter um registro de quem é.
in.fingerd: ALL: (finger @%h | mail -s "finger from %h" root)
```

Qualquer modificação no arquivo */etc/hosts.allow* entrará em ação após reiniciar o *daemon inetd*. Isso pode ser feito com o comando *invoke-rc.d openbsd-inetd restart*.

O arquivo */etc/hosts.deny* é um arquivo de configuração das regras que descreve quais computadores não têm a permissão de acessar um serviço na máquina. Um modelo simples deste arquivo pode ser observado na **Listagem 3**.

Listagem 3. Exemplo do arquivo hosts.deny.

```
# /etc/hosts.deny
#
# Bloqueia o acesso de computadores com endereços suspeitos
ALL: PARANOID
#
# Bloqueia todos os computadores
ALL: ALL
```

A entrada *PARANOID* será redundante, pois a entrada seguinte irá negar tudo. Qualquer uma destas linhas pode aumentar a segurança dependendo de seu requerimento em particular. Tendo um padrão que bloqueia tudo *"ALL*: *ALL"* no arquivo */etc/hosts.deny*, deverão então ser ativados os serviços específicos no */etc/hosts.allow*, permitindo acesso aos computadores que se deseja, e assim deixando uma configuração mais adequada e segura. Qualquer modificação no arquivo */etc/hosts.deny* entrará em ação após reiniciar o *daemon inetd*, pois este é responsável pelas requisições. Isto pode ser feito executando o comando *invoke-rc.d openbsd-inetd restart*, de forma semelhante a *hosts.allow*. Para uma descrição completa das opções fornecidas, deve-se verificar a página de manual apropriada com o comando *man 5 hosts_access*.

O serviço *telnet* utiliza texto plano para seção (exceto nas versões cliente/servidor "-ssl"). Os dados transmitidos por serviços que utilizam texto plano podem ser capturados por *sniffers* (programas que capturam pacotes em uma rede) e trazer perigo ao sistema. É recomendável somente executar o servidor *telnet* padrão em ambientes seguros (como em uma rede interna) e a versão com suporte a *ssl* para fazer conexões via redes inseguras (como a Internet). O serviço *SSH* é uma excelente alternativa ao *telnet*. Além de possuir outras características adicionais que justificam seu uso, possui programas cliente para sistemas operacionais GNU/Linux e Windows.

A falta de segurança no protocolo *telnet* pode ser observada com o uso do aplicativo *dsniff* (uma coleção de ferramentas de análise de conteúdo de pacote TCP/IP, utilizado para capturar o tráfego de rede). Sendo assim, é possível monitorar o tráfego da rede e descobrir problemas como:

- **Tráfego intenso:** se a rede estiver com uma velocidade inferior aos padrões, uma boa alternativa é utilizar um *sniffer* para descobrir as possíveis causas;
- **Serviços de rede consumindo exageradamente recursos do sistema:** um *sniffer* possibilita verificar quais máquinas estão utilizando o serviço. Determina a quantidade de conexões que um cliente estabeleceu com o servidor;
- **Monitoramento de serviços:** muitas vezes é necessário saber como determinados serviços estão se comportando na rede, principalmente os que não possuem um sistema eficiente de *log*.

Um programa pode ser utilizado com o objetivo de capturar informações sigilosas na rede. Este é um dos principais problemas de segurança nas redes de computadores atualmente. O *telnet*, por exemplo, é um tipo de serviço que não utiliza de criptografia para trafegar seus dados na rede, estando assim vulnerável a esse tipo de ataque.

Com o *dsniff* pode-se capturar todos os dados que trafegam na rede, inclusive os *bits* dos pacotes TCP/IP que armazenam as senhas.

No exemplo a seguir, o *host* cliente com IP 192.168.105.7 faz uma conexão *telnet* no servidor com IP 192.168.105.100. Toda operação do cliente será capturada com o uso do *dsniff*, conforme visto na **Listagem 4**.

Listagem 4. Exemplo de captura com dsniff.

```
# dsniff -d -m -i eth0 -w trafego.log
```

Onde:

- *dsniff*: programa utilizado para capturar o tráfego;
- -d: parâmetro usado para modo de debug;
- -m: parâmetro utilizado para detectar automaticamente os protocolos;
- -i : parâmetro utilizado para indicar uma interface;
- eth0: interface de rede utilizada;
- -w: parâmetro utilizado para informar um arquivo de saída;
- *trafego.log*: arquivo onde os dados de saída serão gravados.

Com o uso da opção "-r" após o comando *dsniff*, pode-se analisar os dados capturados, como por exemplo, usuário "flavio" e senha "!@s3curityd3b#2o1o". Mesmo a senha sendo considerada segura, por fazer combinação de letras, números e caracteres especiais, ela foi capturada com sucesso, assim como toda operação efetuada pelo usuário. Observe que o exemplo da saída do comando é visto na **Listagem 5**.

SSH

O *SSH* (*Secure shell*) é conhecido pelos administradores como o canivete suíço da administração de sistemas por possibilitar que toda a configuração, atualizações de segurança e instalação de novos programas seja realizada remotamente em sistemas Linux.

Ele proporciona flexibilidade ao administrador, pois as conexões remotas utilizando *SSH* podem ser realizadas a partir de vários sistemas operacionais clientes, entre eles o Windows, que possui aplicativos capazes de realizar acesso seguro a um sistema GNU/Linux, como exemplo, o aplicativo *Putty*.

O benefício de realizar suporte remoto em ambientes GNU/Linux com segurança, seja em uma LAN (*Local Area Network*) ou

WAN (*Wide Area Network*), é certamente o ponto principal do *SSH*. As técnicas de criptografia empregadas em sua utilização são o que o difere das demais soluções disponíveis e, se bem configurado, é capaz de proporcionar os mais altos níveis de segurança para um acesso remoto, chegando a ser citado por alguns autores como sendo impenetrável virtualmente.

Listagem 5. Exemplo de log capturado pelo dsniff.

```
# dsniff -r trafego.log
----------------------------
07/10/10 08:32:59 tcp 192.168.105.7.1046 -> 192.168.105.100.23 (telnet)
flavio
!@s3curityd3b#2o1o
cd /etc
ls -l
cat fstab
exit
```

Estudos revelam que é inviável tentar quebrar uma senha utilizada por uma conexão *SSH* com força bruta (ver **Nota DevMan 2**), uma vez que esse tipo de conexão utiliza criptografia de 512 bits para a realização de troca de chaves. Pode-se ter uma ideia do tempo que levaria para quebrar uma chave utilizada em uma conexão *SSH* analisando estudos realizados no protocolo de criptografia mais tradicional, o *DES*, que utiliza 64 bits para a criptografia de chaves. É possível decifrar uma chave criptografada com *DES* em menos de um dia, caso se tenha acesso a uma estrutura de cluster com 100 máquinas e processadores *Xeon* ou *Opteron quad-core*.



Nota do DevMan 2

Força Bruta

Consiste na utilização de ferramentas capazes de testar várias combinações possíveis de chaves até conseguir a combinação correta e obter acesso ao sistema.

A vulnerabilidade de segurança do acesso remoto está ligada ao rápido descobrimento das chaves de acesso por exploração de vulnerabilidades existentes nos algoritmos de encriptação. No entanto, vulnerabilidades graves nos algoritmos de encriptação utilizados no *SSH* são descartadas totalmente, uma vez que diversos servidores importantes de todo o mundo utilizam este recurso para sua administração.

O *SSH* utiliza em seu processo de autenticação chaves assimétricas, ou seja, são geradas duas chaves, sendo uma pública (utilizada para encriptar os dados) e outra privada (utilizada para decriptar os dados).

É interessante descrever que, na utilização de chaves assimétricas, a informação é encriptada com a chave pública e decriptada somente pelo detentor da chave privada correspondente. Diferente do processo com chave simétrica, onde é utilizada uma chave única para a encriptação e decriptação dos dados, conforme observado na **Figura 4**.

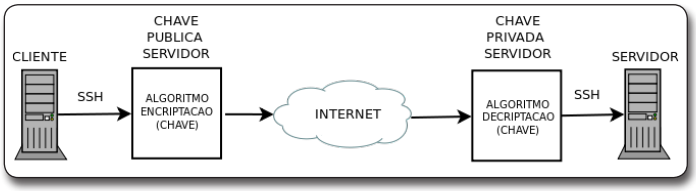


Figura 4. Representação do processo de autenticação realizado no SSH

Inicialmente o cliente *SSH* solicita uma conexão à máquina servidora. O primeiro passo, antes que ocorra qualquer troca de informação entre as máquinas, consiste na troca de suas chaves públicas. Ou seja, o servidor envia ao cliente a sua chave pública, e o cliente envia sua chave pública ao servidor para que seja realizada a checagem de autenticidade durante a troca de chaves e dados, como apresenta a **Figura 5**.

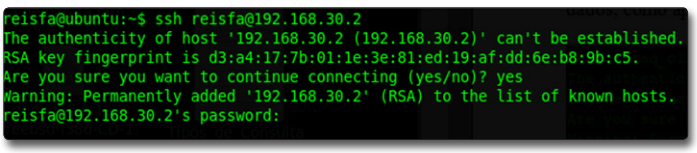


Figura 5. Solicitação de conexão SSH enviada por um cliente

Após a troca de chaves entre os envolvidos, será solicitado usuário e senha para autenticação do acesso. Essa e todas as outras chaves recebidas pelo cliente *SSH* serão armazenadas no arquivo *.ssh/known_hosts* do usuário, onde se encontram as chaves públicas de todos os *hosts* conhecidos por ele. A **Figura 6** apresenta uma chave pública enviada pelo servidor ao cliente.

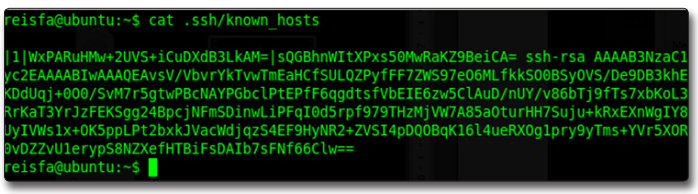


Figura 6. Verificação de persistência de chave pública enviada pelo servidor

Devido à grande aceitação e utilização do serviço de acesso remoto por meio do *SSH*, sua instalação tornou-se bastante simples. Os pacotes "*openssh-server*" e "*openssh-client*" já são encontrados na grande parte dos repositórios das mais variadas distribuições GNU/LINUX, assim como no site padrão do projeto (*http://www.openssh.com/*), onde poderão ser localizadas as últimas atualizações.

Grande parte dos administradores tem como conhecimento que o arquivo tem uma configuração padrão, mas é possível proporcionar diversas configurações de segurança. São essas configurações que serão abordadas a partir de agora.

Hardening de serviço SSH

A realização de uma análise no arquivo de configuração do servidor é fundamental para que o serviço seja disponibilizado com alto nível de segurança. Após uma instalação padrão, algumas configurações podem tornar inseguro o serviço, tais como, permitir *login* remoto por qualquer usuário, inclusive o *root*, o que pode ser visto na **Figura 7**.

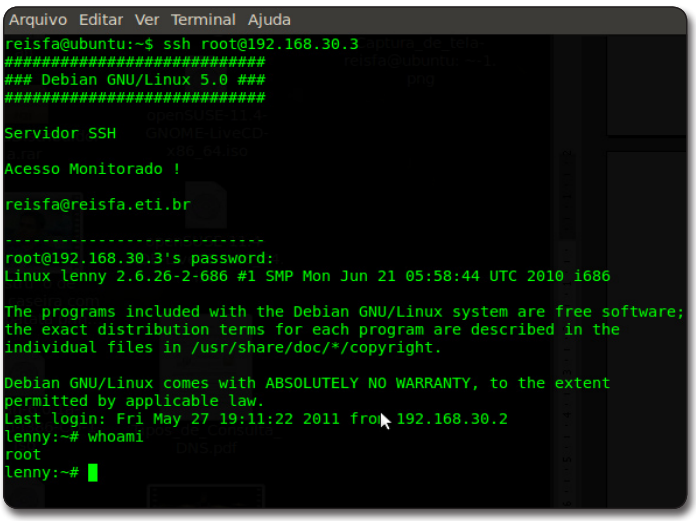


Figura 7. Acessando o servidor SSH como root

As configurações de segurança do servidor *SSH* estão disponíveis no arquivo */etc/ssh/sshd_config*. Vale salientar que no diretório */etc/ssh* também está presente o arquivo de configuração do cliente *SSH*, o "ssh_config".

Para iniciar a configuração do servidor deve-se editar o arquivo */etc/sshd_config*, conforme exemplo da **Listagem 6**.

Por padrão, o servidor *SSH* vem configurado para aceitar conexões na porta 22. É aconselhável que esse valor seja alterado, por exemplo, para 2222 (**Listagem 7**).

Listagem 6. Editando conteúdo do arquivo sshd_config.

```
# vim /etc/ssh/sshd_config
```

Listagem 7. Porta padrão SSH.

```
# Package generated configuration file
# See the sshd(8) manpage for details
# What ports, IPs and protocols we listen for
Port 22
# Use these options to restrict which interfaces/protocols sshd will bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
```

Dessa forma, a maioria das tentativas de ataques como, por exemplo, utilizando *scanners* de rede, não irão obter sucesso em suas varreduras, pois normalmente são configurados para realizar buscas a portas conhecidas e, não encontrando o serviço, dificulta a possibilidade de exploração de alguma brecha de segurança que venha a existir no servidor *SSH*. Após realizar essa alteração, será necessário utilizar o parâmetro "*-p*" no comando de conexão do cliente para explicitar a porta que está sendo utilizada no servidor, de acordo com a **Listagem 8**.

A diretiva "*ListenAddress 0.0.0.0*" permite ao administrador do sistema apontar em quais interfaces o servidor *SSH* poderá ser acessado como, por exemplo, em servidores *firewall*, onde existem duas interfaces de rede. Pode-se informar o endereço IP da interface interna para que somente sejam permitidos acessos via *SSH* originados da rede local, como apresenta a **Listagem 9**.

Por padrão, o servidor *SSH* está habilitado para receber conexões originadas de qualquer uma de suas interfaces.

A diretiva "*Protocol*" representa a versão do protocolo utilizada no servidor. Atualmente os pacotes disponibilizados nos repositórios das distribuições GNU/LINUX já se encontram atualizados para a versão 2, uma vez que a versão *SSH* 1 foi totalmente descontinuada devido a sérios problemas de segurança em seu funcionamento. Portanto, no servidor *SSH* deve somente constar a versão 2 na linha protocolo, conforme a **Listagem 10**.

Listagem 8. Utilização do parâmetro -p para solicitação de acesso remoto.

```
$ ssh -p 2222 flavio@192.168.30.1
```

Listagem 9. Apontando a interface de conexão.

```
ListenAddress 192.168.30.3
```

Listagem 10. Apontando a versão de SSH aceita pelo servidor.

```
Protocol 2
```

Como informado anteriormente, o servidor *SSH*, em sua configuração padrão, permite que o usuário *root* realize acesso remoto ao servidor, o que não é uma boa prática de segurança. Se um invasor obtiver sucesso na exploração de uma vulnerabilidade, este terá acesso total ao sistema e poderá realizar qualquer atividade como instalação, alteração ou remoção de pacotes, utilizar os recursos de hardware para ataques em novas estruturas, entre outros.

Para resolver este problema de segurança, deve-se utilizar a opção "*PermitRootLogin*", encontrada no arquivo de configuração. Por padrão, essa opção recebe o valor "yes" e deve ser trocada para "no", como visto na **Listagem 11.**

Após realizar a alteração de permissão de acesso do usuário *root*, somente usuários comuns poderão acessar o servidor *SSH*. Com isso, é reduzido consideravelmente o risco, uma vez que será necessário realizar o *login* com um usuário comum e, depois de logado, fazer o *login* como *root* utilizando o comando *sudo su -*. Deste modo o invasor terá que descobrir duas senhas antes de ter acesso como administrador ao sistema.

É possível realizar um filtro ainda mais restritivo com a utilização das opções do arquivo de configuração. Por exemplo: pode-se retirar o comentário da opção "*AllowUsers*" e informar como parâmetro o nome do usuário que terá direito de realizar acesso remoto ao servidor *SSH*, como pode ser observado na **Listagem 12**.

Com o objetivo de dificultar as tentativas de invasão do servidor por um intruso que, por ventura, venha a descobrir a porta em que funciona o servidor *SSH*, é disponibilizada a opção "*LoginGraceTime*" que, quando utilizada, emprega o valor informado como parâmetro para limitar o tempo de tentativa de uma conexão. No exemplo da **Listagem 13** está sendo informado o tempo de 60 segundos para a tentativa de conexão. Caso o *login* não seja realizado com sucesso, a conexão é fechada automaticamente.

Listagem 11. Retirando permissão de acesso remoto do usuário root.

```
# Valor padrão
PermitRootLogin yes
# Valor alterado
PermitRootLogin no
```

Listagem 12. Habilitando usuário para realizar acesso remoto.

```
AllowUsers reisfa
```

Listagem 13. Informando tempo para tentativa de conexão.

```
LoginGraceTime 60
```

Em resumo, pode-se concluir que com as configurações realizadas acima, o servidor somente aceitará conexões de clientes *SSH* versão 2, na porta 2222 e que estejam na rede local. Atendendo a essas necessidades iniciais, somente o usuário *marcos* poderá autenticar-se e fazer *login* no sistema.

Recent

O módulo *recent* está disponível para uso com o *iptables*. *Iptables* é um *firewall* em nível de pacotes que funciona baseado no endereço/porta de origem/destino do pacote, prioridade, etc. Ele faz uma comparação de regras para saber se um pacote tem ou não permissão para passar.

O módulo cria uma tabela dinâmica com endereços IPs "recentes", ou IPs setados pelo administrador, podendo dessa forma criar exceções. Um ponto interessante referente a esse módulo é que se pode alterar dinamicamente uma regra que foi anteriormente criada.

O administrador poderá criar regras no *firewall* para manter uma porta fechada e liberá-la quando um usuário mandar uma sequência de pacotes *syn* a algumas portas pré-determinadas. A porta referente ao serviço desejado será mantida fechada e, quando for necessário efetuar o acesso, um pedido de conexão via *telnet* será enviado nas portas definidas.

Assim, a porta será aberta para o IP informado. Se algum invasor tentar executar um *portscanner* não terá sucesso, pois a porta estará fechada.

Port Knocking

O *port knocking* pode ser utilizado como um segredo em uma estrutura de *firewall*, para que assim, o administrador possa liberar o acesso remoto ao servidor. Deste modo, o cliente deverá enviar uma sequência de pacotes para um determinado número de portas com um limite de tempo pré-configurado. Após o envio dos pacotes, o acesso será liberado ao IP de origem. Uma opção muito interessante que pode ser utilizada é a possibilidade de limitar o tempo de acesso para essa conexão.

Para exemplificar o funcionamento do Port Knocking, será criado um cenário utilizando regras do *iptables* para exemplificar a utilização da técnica de *port knocking*. As regras podem ser inseridas diretamente no terminal, mas se a máquina for reiniciada, as regras serão apagadas. Para auxiliar na criação e administração dessas regras, aconselha-se criar um *script* em *shell script*.

As regras aqui propostas têm como finalidade apresentar o uso da técnica de *port knocking*. Nenhuma outra configuração será apresentada neste artigo.

A primeira linha que compõe o script é a "#!/bin/bash", a qual indica o *shell* a ser utilizado para executar as regras. Sempre que um *script* for utilizado para efetuar qualquer tarefa no sistema, o *shell* a ser utilizado deverá ser indicado no início do script, como pode ser visto na **Listagem 14**.

Assim como em qualquer aplicação, o uso de variáveis é utilizado para facilitar a manutenção do script. A primeira variável, "v_CMD_IPTABLES", guarda a localização do binário do comando *iptables*, retornada pelo comando "*which*". Dessa forma, independente do local que este seja disponibilizado, o sistema poderá localizá-lo. A segunda variável, "v_REDE", foi utilizada para armazenar a rede a qual será utilizada para liberar o acesso. E, por último, a variável "v_IP_LIBERADO", onde é armazenado o IP do *host* do administrador, o qual não irá passar pela regra criada. Observa-se na **Listagem 15** a criação das variáveis.

Prosseguindo na configuração do *firewall*, devem-se zerar quaisquer regras existentes. Dessa forma, o sistema não correrá o risco de executar regras que não estejam previstas. Para esse fim, utilizam-se os parâmetros "-F" (de *flush*) para limpar as regras, "-X" para eliminar as *chains* extras existentes (veja a **Nota DevMan 3**), e o "-Z" para zerar todos os contadores, observe na **Listagem 16**.

Listagem 14. Indicando o shell que será utilizado para carregar as regras.

```
#!/bin/bash
```

Listagem 15. Criando as variáveis utilizadas.

```
v_CMD_IPTABLES=$(which iptables);
v_REDE="192.168.30.0/29";
v_IP_LIBERADO="192.168.30.2"
```

Listagem 16. Limpando possíveis regras existentes.

```
$v_CMD_IPTABLES -F
$v_CMD_IPTABLES -X
$v_CMD_IPTABLES -Z
```



Nota do DevMan 3

Chain

São locais onde as regras do firewall definidas pelo usuário são armazenadas para operação do firewall. Existem diferentes tipos de chains: os embutidos (como as chains INPUT, OUTPUT e FORWARD) e os criados pelo usuário. Os nomes das chains embutidos devem ser especificados sempre em maiúsculas (note que os nomes das chains são case sensitive, ou seja, o chain input é completamente diferente de INPUT).

Sempre que um *firewall* é criado uma política padrão deverá ser criada. Pode-se citar duas situações, uma política onde tudo é

permitido, exceto o que é expressamente proibido. Nessa forma, a *chain INPUT* é responsável pelos dados de entrada no *firewall*, *OUTPUT* é responsável pelos dados de saída do *firewall* e *FORWARD* manipula os dados que passam pelo *firewall* que receberam seu alvo como *ACCEPT*. Posteriormente deverão ser criadas regras de exceção. Outra forma é criar uma política restritiva, bloqueando todo o tráfego inserindo o alvo *DROP* para todas as três *chains* e, assim, liberar somente o que for necessário. No exemplo utilizado neste artigo será adotada a segunda opção, configurando as três *chains* como *DROP* (**Listagem 17**).

Na regra seguinte do *firewall*, serão verificadas as requisições de entrada (pode-se observar que é usada a *chain INPUT*), quando chegar um pacote de nova conexão (*-m state NEW*) vindo da interface de rede interna (*-i eth0*) com destino à porta 42129 (*--dport 42129*), o parâmetro (*-m recent*) indica que o módulo *recent* será ativado podendo assim analisar as tabelas criadas. Caso não exista a tabela, a mesma será criada com o nome *sshlimita* (*--name sshlimita*) e irá retornar para a tabela *INPUT* analisando as próximas regras (*-j RETURN*). Observa-se um exemplo da regra na **Listagem 18**.

Listagem 17. Definindo a Política Padrão do Firewall.

```
$v_CMD_IPTABLES -P OUTPUT DROP
$v_CMD_IPTABLES -P INPUT DROP
$v_CMD_IPTABLES -P FORWARD DROP
```

Listagem 18. Verificando estado da nova conexão.

```
$v_CMD_IPTABLES -A INPUT -p tcp --dport 42129 -i eth0 -m state --state NEW –
 m recent --set --name sshlimita -j RETURN
```

A próxima regra irá verificar as requisições de entrada (*-A INPUT*). Quando chegar um pacote de nova conexão (*-m state NEW*) vindo da interface de rede interna (*-i eth0*) com destino à porta 42129 tcp (*-p tcp --dport 42129*), o parâmetro (*-m recent*) indica que o módulo *recent* será ativado e irá verificar (*--rcheck*) se na tabela *sshlimita* (*--name sshlimita*) existe a entrada do IP de origem. Se a resposta for verdadeira, o módulo irá verificar se este tem três entradas (*--hitcount 3*) em menos de dois minutos (*--seconds 120*), caso contrário, o firewall irá descartá-la. Caso a resposta seja positiva, para fins de auditoria, o *firewall* irá adicionar uma entrada de nome "*SSHLimitaConn*" no arquivo de *log* de mensagens do *kernel*, e o administrador poderá dessa forma acompanhar os registros nos arquivos de *log*. O *syslog* é uma das ferramentas disponíveis no GNU/Linux que podem ser utilizadas para auxiliar a armazenagem de *logs* e auditoria dos mesmos. Sua configuração pode ser encontrada em */etc/syslog.conf*. Um exemplo da segunda regra do *iptables* pode ser observado na **Listagem 19**.

A terceira regra verifica as requisições na tabela de entrada (*-A INPUT*) quando chegar um pacote de nova conexão (*-m state NEW*) vindo da interface da rede interna (*-i eth0*) com direção à porta 42129 tcp (*-p tcp --dport 42129*). O módulo *recent* será ativado (*-m recent*) e será verificado (*--rcheck*) se na tabela *sshlimita* (*--name sshlimita*) existe a entrada do *IP* de origem. Caso a resposta seja positiva, será verificado se tem três entradas (*--hitcount 3*) em um intervalo menor que dois minutos (*--seconds 120*). Se a resposta for positiva, a tentativa de conexão será negada (*-j DROP*), conforme visto na **Listagem 20**.

O administrador poderá optar por liberar sua máquina do segredo aplicado nas regras. Essa opção pode ser feita por IP ou até mesmo por rede. No exemplo aplicado no artigo foi liberado o IP do administrador, aqui representado pela variável "$v_IP_LIBERADO". Caso o administrador venha a trocar de endereço IP, basta que o mesmo altere o valor atribuído à variável no início do script. A linha inserida no *script* pode ser observada na **Listagem 21**.

As conexões com o *firewall* devem ser estabilizadas a fim de manter a conectividade entre cliente e servidor. Para que isso ocorra, uma regra é adicionada estabilizando as conexões aceitas pelo *firewall*. Um exemplo pode ser observado na **Listagem 22**.

Listagem 19. Logando tentativas de Brute Force.

```
$v_CMD_IPTABLES -A INPUT -p tcp --dport 42129 -i eth0 -m state --state NEW
-m recent --name sshlimita --rchek --seconds 120 --hitcount 3 -j LOG --log-prefix
"SSHLimitaConn: "
```

Listagem 20. Bloqueando tentativas de Brute Force.

```
$v_CMD_IPTABLES-A INPUT -p tcp --dport 42129 -i eth0 -m state --state NEW -m
recent --name sshlimita --rcheck --seconds 120 --hitcount 3 -j DROP
```

Listagem 21. Liberando acesso para máquina do Administrador.

```
$v_CMD_IPTABLES -A INPUT -p tcp -s $v_IP_LIBERADO --dport 42129 -j ACCEPT
```

Listagem 22. Estabilizando as conexões.

```
$v_CMD_IPTABLES -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Na **Listagem 23** observam-se três conjuntos de regras. Na primeira linha de cada um dos conjuntos utiliza-se o parâmetro "iptables –N", onde cada nova *chain* será inserida com o nome "PORTKNOCK-FASE2", "PORTKNOCK-FASE3" e "PORTKNOCK-FASE4", respectivamente. Dessa forma, o administrador pode acompanhar e organizar todo o processo de acesso ao servidor, utilizando as portas posteriormente definidas no *firewall*. Na segunda e terceira linhas, faz-se a alteração das fases, removendo a fase atual e indicando a próxima fase. Na quarta linha é efetuado um *log* desse processo. Na última listagem desta matéria é apresentado um registro do arquivo de *log* dessas regras, facilitando assim o entendimento da alteração entre as fases.

Quando o pacote passa por todas as regras, o módulo *recent* fará um *update* na *chain* retornando para a FASE1, responsável por analisar a primeira porta da sequência escolhida. Um exemplo da regra pode ser visto na **Listagem 24**.

O próximo passo da criação das regras do *firewall* será determinar quais portas serão controladas e também qual será o intervalo, em segundos, que cada uma das fases irá aguardar até que a próxima porta seja indicada para acertar o segredo. No cenário proposto, será utilizado um intervalo de 15 segundos entre cada fase, e caso o tempo expire, o processo deverá ser iniciado novamente. Essas regras podem ser observadas na **Listagem 25**.

Listagem 23. Criando as fases do PortKnocking.

```
$v_CMD_IPTABLES -N PORTKNOCKING-FASE2
$v_CMD_IPTABLES -A PORTKNOCKING-FASE2 -m recent --name FASE1 --remove
$v_CMD_IPTABLES -A PORTKNOCKING-FASE2 -m recent --name FASE2 --set
$v_CMD_IPTABLES -A PORTKNOCKING-FASE2 -j LOG --log-prefix "INTO FASE2: "

$v_CMD_IPTABLES -N PORTKNOCKING-FASE3
$v_CMD_IPTABLES -A PORTKNOCKING-FASE3 -m recent --name FASE2 --remove
$v_CMD_IPTABLES -A PORTKNOCKING-FASE3 -m recent --name FASE3 --set
$v_CMD_IPTABLES -A PORTKNOCKING-FASE3 -j LOG --log-prefix "INTO FASE3: "

$v_CMD_IPTABLES -N PORTKNOCKING-FASE4
$v_CMD_IPTABLES -A PORTKNOCKING-FASE4 -m recent --name FASE3 --remove
$v_CMD_IPTABLES -A PORTKNOCKING-FASE4 -m recent --name FASE4 --set
$v_CMD_IPTABLES -A PORTKNOCKING-FASE4 -j LOG --log-prefix "INTO FASE4: "
```

Listagem 24. Atualizando as fases do PortKnocking.

```
$v_CMD_IPTABLES -A INPUT -m recent --update --name FASE1
```

Listagem 25. Conferindo a sequência de portas do PortKnocking.

```
$v_CMD_IPTABLES -A INPUT -p tcp --dport 12345 -m recent --set --name FASE1
$v_CMD_IPTABLES -A INPUT -p tcp --dport 51234 -m recent --rcheck --seconds 15
--name FASE1 -j INTO-FASE2
$v_CMD_IPTABLES -A INPUT -p tcp --dport 45123 -m recent --rcheck --seconds 15
--name FASE2 -j INTO-FASE3
$v_CMD_IPTABLES -A INPUT -p tcp --dport 34512 -m recent --rcheck --seconds 15
--name FASE3 -j INTO-FASE4
```

Dica DevMan

Não é aconselhável utilizar portas em sequências crescentes ou decrescentes, assim como utilizar portas próximas, como: 400, 410, 420 e 430. O ideal é intercalar os valores evitando que algum portscan consiga identificar a sequência.

A última fase da configuração do *firewall* será liberar a conexão com a porta 42129 (*SSH*) por um tempo de 1800 segundos (30 minutos) somente para pacotes vindos da rede armazenada na variável "v_REDE". Decorrido esse tempo, o acesso será interrompido para o IP em questão, sendo necessário um novo pedido de conexão. É importante destacar que, mesmo que o usuário ainda esteja logado no sistema, a conexão será interrompida. Um exemplo desta regra pode ser visto na **Listagem 26**.

A última regra aplicada ao firewall irá atuar nas conexões que não passaram pelo segredo. Desse modo, todas as requisições feitas na porta 42129 serão negadas por essa regra. Aqui também se encerram as configurações no servidor. Pode-se observar um exemplo na **Listagem 27**.

Para que se possa ter certeza de que a porta do servidor destinado ao serviço de SSH está fechada para novas conexões, pode-se executar uma consulta no servidor utilizando um *sniffer* de rede para capturar todas as portas abertas. *Nmap* é um *software* livre que realiza um escaneamento de portas. Ele é utilizado para avaliar a segurança dos computadores e para descobrir serviços ou servidores em uma rede de computadores. Analisando o resultado da varredura feita com o *Nmap*, pode-se verificar que a porta 42129 está configurada para receber conexões e está sendo filtrada. Observa-se um exemplo simples de uso do *Nmap* na **Listagem 28**.

Efetuando os testes

Para que se possa efetuar os testes de acesso no servidor, será empregado o aplicativo *Knock*, utilizado para efetuar conexões com intervalos pré-determinados, enviando pacotes às portas configuradas no intervalo de tempo escolhido. Para este teste foi utilizado um *host Ubuntu* como cliente e um servidor GNU/Linux Debian Lenny.

Para isso, o primeiro passo é observar se o aplicativo *knock* está disponível no *host Ubuntu*. Para conferir sua instalação, pode-se utilizar o comando "*dpgk –l | grep knock*". Se nada for retornado, usa-se o comando "*aptitude install knock*" para instalar a aplicação.

Para facilitar o teste, será criado um pequeno script em *shell*. O script deverá iniciar indicando o *shell* que será utilizado, neste caso, "#!/bin/bash". Na sequência do *script*, o aplicativo *knock* recebe como parâmetro o IP do servidor e as portas do segredo na sequência em que foram definidas no *firewall*. Um intervalo de três segundos é adicionado com o uso do comando "*sleep*" e, logo em seguida, é solicitada a conexão via *ssh*. Para executar a aplicação deve-se atribuir a permissão de execução com o comando *chmod*. O código do *script* é apresentado na **Listagem 29**.

Listagem 26. Autorizando conexões.

```
$v_CMD_IPTABLES -A INPUT -p tcp -s $v_REDE --dport 42129 -m recent --rcheck
--seconds 1800 --name FASE4 -j ACCEPT
```

Listagem 27. Recusando todas as demais requisições na porta 42129.

```
$v_CMD_IPTABLES -A INPUT -p tcp --dport 42129 -j DROP
```

Listagem 28. Comando sugerido para testes.

```
$ nmap 192.168.30.3 -p 42129

Starting Nmap 5.00 ( http://nmap.org ) at 2010-07-10 11:49 BRT
Interesting ports on 192.168.30.3:
PORT     STATE    SERVICE
42129/tcp filtered unknown
```

Listagem 29. Script sugerido para testes.

```
#!/bin/bash
knock 192.168.30.3 12345:tcp 51234:tcp 45123:tcp 34512:tcp
sleep 3
ssh –p 42129 reisfa@192.168.30.3
```

Durante a configuração do *firewall* no servidor, algumas regras foram utilizadas para registros de *log*. Esses registros podem ser acompanhados no arquivo */var/log/messages*. Nota-se em destaque na **Listagem 30** a origem do pacote em "*SRC*=192.168.30.2", ou seja, onde foi executado o aplicativo *Knock*, o cliente *Ubuntu*. Em "*DST=192.168.30.3*" observa-se o IP de destino do pacote, no exemplo o IP do servidor Debian. O "*DPT=51234*" é a primeira porta configurada no servidor. A opção *SYN* (*synchronize*) indica uma solicitação para que uma conexão seja estabelecida. As três linhas do *log* coletadas e apresentadas indicam as portas escolhidas no servidor, onde pode ser observada a diretiva DPT (51234, 45123, 34512), fechando assim o ciclo completo para disponibilizar a conexão.

Listagem 26. Autorizando conexões.

```
$v_CMD_IPTABLES -A INPUT -p tcp -s $v_REDE --dport 42129 -m recent --rcheck
--seconds 1800 --name FASE4 -j ACCEPT
```

Listagem 27. Recusando todas as demais requisições na porta 42129.

```
$v_CMD_IPTABLES -A INPUT -p tcp --dport 42129 -j DROP
```

Listagem 28. Comando sugerido para testes.

```
$ nmap 192.168.30.3 -p 42129

Starting Nmap 5.00 ( http://nmap.org ) at 2010-07-10 11:49 BRT
Interesting ports on 192.168.30.3:
PORT     STATE    SERVICE
42129/tcp filtered unknown
```

Listagem 29. Script sugerido para testes.

```
#!/bin/bash
knock 192.168.30.3 12345:tcp 51234:tcp 45123:tcp 34512:tcp
sleep 3
ssh –p 42129 reisfa@192.168.30.3
```

Listagem 30. Log de mensagem do kernel /var/log/messages.

```
Jul  6 02:11:13 debian kernel: [ 1824.496437] PORTKNOCK-FASE2: IN=eth0 OUT= M
AC=08:00:27:54:e9:c6:00:1c:c0:1c:e4:96:08:00 SRC=192.168.30.2 DST=192.168.30.3
LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=18774 DF PROTO=TCP SPT=49872
DPT=51234 WINDOW=5840 RES=0x00 SYN URGP=0

Jul  6 02:11:13 debian kernel: [ 1824.496437] PORTKNOCK-FASEE3: IN=eth0
OUT= MAC=08:00:27:54:e9:c6:00:1c:c0:1c:e4:96:08:00 SRC=192.168.30.2
DST=192.168.30.3 LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=3402 DF PROTO=TCP
SPT=57478 DPT=45123 WINDOW=5840 RES=0x00 SYN URGP=0

Jul  6 02:11:13 debian kernel: [ 1824.496437] PORTKNOCK-FASE4: IN=eth0 OUT= M
AC=08:00:27:54:e9:c6:00:1c:c0:1c:e4:96:08:00 SRC=192.168.30.2 DST=192.168.30.3
LEN=60 TOS=0x00 PREC=0x00 TTL=64 ID=22711 DF PROTO=TCP SPT=49287
DPT=34512 WINDOW=5840 RES=0x00 SYN URGP=0
```

Conclusão

Neste artigo foram apresentadas técnicas de conexões remotas, destacando algumas de suas vulnerabilidades, como no exemplo de uso do *telnet*, onde todos os seus dados trafegam em texto puro, sem criptografia, sendo assim facilmente capturados por um *sniffer*. Como substituto ao uso do *telnet*, foi apresentado o SSH (*Secure Shell*), e mesmo com toda segurança inserida no protocolo devido ao uso da criptografia, técnicas de *hardening* foram utilizadas para aumentar ainda mais seu nível de confiabilidade e segurança.

Manter uma aplicação segura em um servidor é uma tarefa importante e muitas vezes negligenciada pelos administradores de redes. Com o uso de *firewalls*, é possível criar uma barreira entre o cliente malicioso e o servidor. E com o módulo *recent* do *iptables* foi possível combinar regras e apresentar a técnica conhecida como *port knocking*, criando um segredo para que a conexão fosse liberada.

Um *cracker* poderá até usar um *scanner* para identificar se há portas abertas no servidor, porém, para que obtenha o acesso, deverá saber a combinação das portas, o que irá dificultar que a tentativa de invasão obtenha sucesso. No entanto, mesmo adotando esses recursos, recomenda-se avaliar os *logs* com frequência, identificando assim possíveis varreduras em busca de falhas na configuração do *firewall*. Além disso, aconselha-se o uso de um IDS (*Snort*) e um IPS (*Guardian*) junto com o *firewall* para aumentar a proteção do perímetro.



Flávio Alexandre dos Reis
reis.falexandre@gmail.com – www.reisfa.eti.br
Especialista em Segurança da Informação – FMG/JF, Pós Graduando em Redes de Computadores – CES/JF. Certificado LPIC-1. Atualmente professor em Instituições de Ensino Superior da Região de Juiz de Fora em disciplinas ligadas a redes de Computadores.



Eduardo Pagani Julio
epagani@gmail.com
Mestre em Computação pela Universidade Federal Fluminense. Atualmente professor em Instituições de Ensino Superior da Região de Juiz de Fora em disciplinas ligadas a redes, e consultor atuando principalmente nas áreas de redes de computadores e segurança, com enfoque em Linux.

Links

Site sobre o iptables
www.netfilter.org

Debian Administration
www.debian-administration.org

Site telnet
www.telnet.org

Site SSH
www.openssh.com

Guia Foca Linux
http://focalinux.cipsga.org.br/guia/avancado/ch-fw-iptables.htm

Site do Shorewall
http://www.shorewall.net/PortKnocking.html

Site Zago
http://www.zago.eti.br/ssh/knock.html#toc7

Site da ferramenta dsniff
http://monkey.org/~dugsong/dsniff/

Política de Segurança da Informação e Firewall

Paulo Moreira

Resumo DevMan

De que se trata o artigo:

Neste artigo veremos a importância de integrar um Firewall com uma Política de Segurança da Informação. Para isso, será explicado o que é, qual a funcionalidade e a importância de cada um.

Para que serve:

Uma Política de Segurança da Informação bem definida é a base para garantir o correto uso da informação. A soma dessa política com as políticas de um Firewall serve de apoio ao responsável pela Segurança da Informação da organização junto à alta direção.

Em que situação o tema é útil:

Considerando a informação como o maior ativo de uma empresa, pode-se dizer que essa integração entre Firewall e Política de Segurança da Informação serve para proteger os dados que serão acessados, impedindo que os façam de forma indevida.

Considerando o cenário globalizado atual onde os negócios são suportados cada vez mais pela tecnologia, é de extrema importância requisitos mínimos de segurança da informação, para que informações estratégicas não sejam disseminadas de forma indevida, o que pode impactar de forma significante nos negócios da organização.

Para que essa disseminação dos dados não aconteça de forma indesejada, é importante a adoção de mecanismos de segurança da informação voltados aos processos, pessoas e tecnologia. Dois deles podemos dizer que são essenciais: uma política de segurança de informação, direcionada na sua essência às pessoas e um firewall, recurso provedor de segurança no meio tecnológico, sendo configurado com base na política de segurança da informação.

Neste contexto, esse artigo tem como objetivo discutir como as políticas de segurança da informação podem ser usadas como uma parte ativa do esforço de uma organização para proteger os seus ativos de informações valiosas, juntamente com uma ferramenta importante e indispensável como o firewall.

Segurança da Informação

A Segurança da Informação tem como objetivo proteger as informações de uma determinada pessoa ou empresa, ou seja, não é uma preocupação somente corporativa. Mas à medida que as empresas geram informações, armazenam e as distribui para gerir seus negócios, aumentam os riscos a que os dados ficam expostos.

A segurança de uma determinada informação pode ser afetada por fatores comportamentais e uso de quem se utiliza dela, pelo ambiente computacional que a cerca ou por pessoas mal intencionadas que têm o objetivo de furtar, destruir ou modificar tal informação.

Garantir a proteção da informação e mitigar os riscos, não são somente desafios técnicos, de responsabilidade de analistas e engenheiros, mas desafios que envolvem a gestão como um todo.

Os princípios básicos da segurança da informação que atualmente norteiam na análise, planejamento e na implementação da segurança dos dados, são:

- **Confidencialidade:** consiste em garantir que apenas as pessoas que estão autorizadas a ter acesso à informação são capazes de fazê-lo;
- **Integridade:** garantia de que as informações serão protegidas contra alterações não autorizadas e mantidas a exatidão das mesmas, tal qual como foi armazenada e disponibilizada;
- **Disponibilidade:** consiste em assegurar que os sistemas de informação e a informação estarão disponíveis e operacionais quando necessários, apoiando, assim, os processos de negócios;
- **Autenticidade:** garantia de que a informação é verdadeira, de fonte segura e não sofreu alterações em seu percurso;
- **Não repúdio:** garantia de que o emissor de algum dado ou informação não possa, posteriormente, negar que tenha enviado e/ou alterado alguma informação.

Os benefícios evidentes são reduzir os riscos com vazamentos, fraudes, erros, uso indevido, sabotagens, roubo de informações e diversos outros problemas que possam comprometer estes princípios básicos.

A segurança visa também aumentar a produtividade dos usuários através de um alinhamento estratégico dos controles e recursos de segurança aos requerimentos do negócio, provendo um ambiente mais organizado, onde exista um maior controle sobre os recursos de informática e finalmente, viabilizando aplicações críticas da empresa.

Uma política de segurança bem elaborada (com base no negócio), adequada gestão de risco, a implantação de uma arquitetura de sistemas segura, complementadas com o apoio dos diversos níveis da organização, são fundamentais para garantir uma proteção eficiente para organização.

Políticas de Segurança

Em poucas palavras, uma PSI – Política de Segurança da Informação seria como um documento que é criado para impor

regras ou regulamentos específicos, manter uma estrutura de procedimentos e um conjunto de mecanismos através dos quais os seus objetivos de segurança da informação podem ser definidos e atingidos.

O objetivo de uma política de segurança da informação é prover uma orientação e apoio da direção para a segurança da informação, de acordo com os requisitos do negócio e com as leis e regulamentações pertinentes (ABNT, 2005).



Nota do DevMan

Gestão de Risco: Forma de gerenciar processos de forma contínua, acompanhando os níveis de riscos, a fim de adotar controles para banir vulnerabilidades e mitigar ameaças.

Mais do que isso, tem como objetivo (e talvez o principal) definir padrões de acesso às informações que sejam conhecidos por todos os colaboradores da empresa, com apoio da alta administração. Através desse documento, todos têm o conhecimento das normas impostas pela organização. Quando digo "todos" quero dizer não somente os funcionários daquela empresa, mas também clientes, fornecedores e terceiros.

A política de segurança da informação deve ir além dos recursos computacionais, de forma a abranger todas as áreas de negócios da empresa. Podemos dizer, ainda, que a PSI deva ir além da informação, seja esta armazenada eletronicamente, impressa, etc. Deve-se aliar conhecimento à comunicação e integração das informações, para que o ambiente corporativo, de fato, possa manter a segurança da informação.

Uma PSI normalmente contém princípios éticos e legais a serem atendidos no âmbito da tecnologia, continuidade de negócios, punição aos que infringem a norma estabelecida pela PSI e planos de conscientização.

Uma política de segurança da informação poderá estabelecer que o uso da internet, por exemplo, seja exclusiva para contribuição do negócio, a utilização do correio eletrônico seja apenas para troca de informações de fins profissionais, as informações/documentos devam ter uma classificação quanto a sua importância, novos colaboradores deverão conhecer e assinar um termo de confidencialidade e sigilo nas informações confidenciais da empresa. Enfim, uma política de segurança da informação determina como as pessoas devem manipular e interagir com as informações da organização.

Antes que a política de segurança seja elaborada é importante definir quais as informações que serão protegidas. Para isso, é feita uma análise de riscos que identifica quais os recursos que serão protegidos pela política, ameaças às quais estes recursos estão expostos e vulnerabilidades que podem abrir "portas" para essas ameaças.

Além de uma análise para identificar o que será protegido, é importante que a PSI seja de fácil leitura e entendimento, clara e objetiva, com as responsabilidades definidas e penalidades caso a política seja desrespeitada.

A política de segurança deve ter implantação realista, e definir claramente as áreas de responsabilidade dos usuários, do pessoal de gestão de sistemas, redes e da direção. Deve também adaptar-se a alterações na organização. As políticas de segurança definem procedimentos de segurança adequados, processos de auditoria à segurança e estabelecem uma base para procedimentos legais na seqüência de ataques, refletindo sempre os controles e requisitos necessários ao negócio da empresa.

Não podemos deixar de mencionar, ainda, que cada empresa tem sua regra de negócio, seu ambiente computacional e pessoal. Esses fatores devem ser levados em conta para que a PSI seja elaborada.

Algumas normas definem aspectos que devem ser levados em consideração ao elaborar políticas de segurança. Entre essas normas estão a BS 7799 (elaborada pela British Standards Institution) e a NBR ISO/IEC 17799 (a versão brasileira desta primeira). A ISO começou a publicar a série de normas 27000, em substituição à ISO 17799 (e, por conseguinte à BS 7799), das quais a primeira, ISO 27001, foi publicada em 2005, tendo o seu foco no sistema de gestão da segurança da informação, e a ISO 27002 nos controles requeridos à manutenção desse sistema.

Um dos fatores críticos para o sucesso de uma política de segurança da informação é a conscientização dos usuários, pois através dela os usuários tomam conhecimento das diretrizes, suas responsabilidades e sanções cabíveis ao não cumprimento dos controles de segurança necessários ao negócio. A falta de uma política de segurança da informação impressa, ou publicada, é o mesmo como não tendo qualquer política. Uma publicação e distribuição amplamente da PSI impede que o assunto se distorça dentro da organização. O trabalho de conscientizar geralmente é apoiado por áreas que possuam conhecimento e capacitação para treinamento e publicidade, como os departamentos de Recursos Humanos/Treinamento e Marketing. É importante que cada área envolvida saiba, de forma clara, suas responsabilidades.

Firewall

Firewall é um dispositivo que funciona como corta-fogo, um sistema que impõe uma política de controle de acesso entre duas redes, permitindo ou denegando as transmissões de uma rede à outra. Um uso típico é situá-lo entre uma rede local e a rede Internet, como dispositivo de segurança para evitar que os intrusos possam acessar uma informação confidencial, conforme a **Figura 1**.

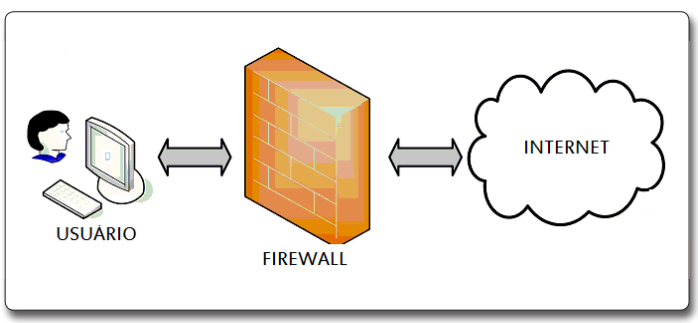


Figura 1. Exemplo de uso de firewall

Pode-se dizer, ainda, que firewall é um componente, ou um conjunto deles, que atua entre uma ou mais redes, por onde passa todo o tráfego. Dependendo do serviço, o firewall decide se permite ou não a conexão. Além disso, o firewall examina se a comunicação está entrando ou saindo, e dependendo da sua direção, pode permiti-la ou não. Deste modo, um firewall pode permitir de uma rede local para a Internet, serviços de web, correio, FTP, etc.

O firewall é um dos elementos mais importantes e conhecidos em ambientes de computadores. Mas, por ser importante, não pode ser considerado o único responsável pela segurança de informação de uma organização, pois não são infalíveis. Não basta apenas instalá-lo como uma única fonte de defesa da rede. O firewall deve ser visto como mais uma ferramenta dentro do seu sistema de segurança da informação.

A utilização de um firewall na rede pode trazer importantes benefícios, como por exemplo, proteção a serviços, acesso controlado, privacidade aumentada, logs de acessos, cumprimento da política de segurança da informação. Esse último, não menos importante, uma vez que o firewall pode ajudar na implantação de uma PSI, sem esperar que os usuários a cumpra.

Mas tudo tem vantagens e desvantagens. No caso de um firewall não é diferente. Como desvantagens podemos citar, por exemplo, que ele não protege sua rede interna, não há proteção contra vírus, não prevê novos tipos de ataques, se o servidor ou *appliance* falhar, sua segurança fica comprometida (a não ser que você tenha uma política de *disaster recovery* para esse incidente).



Nota do DevMan

Disaster recovery: Processo, política ou procedimento que tem como propósito a recuperação ou a continuação da infraestrutura tecnológica fundamental para a organização, após um desastre de origem natural ou fator humano.

Todo firewall contém regras pré-definidas que permitem:
- Autorizar a conexão (allow);
- Bloquear a conexão (deny);
- Rejeitar o pedido de conexão sem avisar o emissor (drop).

Podemos chamar o conjunto dessas regras como sendo a política do firewall, sendo conhecidos dois tipos de políticas:
- *Tudo o que não é expressamente permitido é proibido:* nesse tipo, a política adotada é a **proibitiva**, onde é descrito exatamente o que é permitido acessar;
- *Tudo o que não é expressamente proibido é permitido:* menos comum, a política **permissiva** permite todos os serviços por *default*, com exceção dos serviços que a política de acesso tenha identificado como não permitida.

O modelo de política proibitiva é mais seguro e, também, comumente adotado pelas empresas. Já o modelo permissivo certamente não se enquadra em um modelo seguro, pois o que deve ser proibido é bem maior que os itens permitidos.

Tipos de Firewall

Os firewalls podem ser divididos em quatro tipos:
- Firewall pessoal (não podemos esquecê-lo!);
- Filtro de pacotes;
- Proxy Servers;
- Gateway de Aplicação.

Detalharemos abaixo cada um desses tipos.

Firewall pessoal

Trata-se de uma camada de software inclusa no sistema operacional ou instalada, um software. Ao invés de trabalhar limitando a comunicação entre domínios de segurança, a aplicação limita a comunicação para determinados hosts.

Filtro de Pacotes

É o tipo mais antigo e comum. Ele é colocado entre uma rede confiável e outra não confiável. Opera na camada de Rede e Transporte. Toma suas decisões baseado no endereço IP e nos campos dos pacotes. Permite ou proíbe o forward do pacote usando informações contidas nos Headers dos pacotes como: Endereço IP fonte ou de destino, portas TCP/UDP de destino ou de origem, por protocolos (ICMP, TCP, UDP, SNMP, etc.), entre outros.

O filtro de pacotes pode ser dividido em três tipos:
- **Filtragem Estática:** Implementado na maioria dos roteadores. Suas regras devem ser alteradas manualmente;
- **Filtragem Dinâmica:** Modifica suas regras de acordo com o comportamento dos tráfegos. Por exemplo, só faz o forward de conexões ssh caso elas tenham sido iniciadas por um host dentro da rede ou ainda, só permite pacotes ICMP type 8 (*echo request*) a uma taxa de três pacotes por segundo, passando disso ele nega o tráfego desse protocolo;
- **Filtragem por Status:** Muito semelhante à filtragem dinâmica, só que agrega uma profundidade maior na análise do pacote. Um exemplo desse tipo seria permitir o forward entre hosts dependendo do status da conexão, permitir conexões apenas com status de ESTABLISHED e RELATED.

Proxy Server

É um servidor que é implantado entre a aplicação de um cliente (arquitetura cliente/servidor) e a internet, como por exemplo, um navegador web. Serve como ponte de acesso para a internet ou da internet. É feita uma conexão entre o cliente e o proxy, e outra entre o proxy e o servidor desejado. Pode ser utilizado para registrar o uso da internet e também para bloquear o acesso a um site da Web.

Gateway de Aplicação

Provê o nível mais alto de segurança, atuando na camada de aplicação TCP-IP, aplicando maior segurança em aplicações específicas, como por exemplo, serviços de FTP, Telnet, SNMP, garantindo que não existam conexões entre hosts externos e hosts internos. O gateway de aplicação pode fornecer, sim, um maior

grau de proteção do que o filtro de pacote, porém, apesar de eficiente, essa técnica pode degradar a performance da rede.

Integração

Firewalls são a primeira defesa de qualquer rede contra intrusos da internet ou entre redes distintas. No entanto não devem ser tratados como ponto único em um sistema de segurança da informação, mas sim, parte dele.

Não podemos esquecer o básico: um bom Sistema de Detecção de Intrusos (IDS), um Sistema de Prevenção de Intrusos (IPS), um bom antivírus e uma DMZ bem configurada. Contudo, antes de todas essas ferramentas, é importante a concepção de uma Política de Segurança da Informação bem planejada, definida, divulgada e, na medida do possível, revisada de tempos em tempos.



Nota do DevMan

DMZ: Em segurança de computadores, uma DMZ, zona desmilitarizada ou ainda rede de perímetro, é uma rede física ou lógica que contém serviços (HTTP, FTP, e-mail) que possuem acesso externo, situada entre uma rede confiável e uma rede não confiável (rede local e internet).

A complexidade dos sistemas de segurança que citamos, em geral, é cada vez maior. A integração desses sistemas faz com que a empresa chegue a um "produto" final: garantir a segurança dos dados naquela organização. O trabalho em conjunto de um firewall com uma política de segurança da informação vai de encontro com esse produto.

Conclusão

O tema segurança da informação mostra-se altamente abrangente, reunindo não somente áreas da informática, mas também disciplinas como análise de riscos, treinamento, dentre outras. Alia gestão e planejamento da informação, além de mecanismos sociais e tecnológicos.

Com a complexidade e novas tecnologias surgindo a todo o momento, faz-se necessário para qualquer organização a criação de uma política de segurança formal, clara, objetiva e publicada, ou seja, a política deve estar disponível e ser de conhecimento de todos os colaboradores, para que efetivamente seja praticada no dia-a-dia corporativo. A política definida e adotada pela organização deve estar sempre ajustada com o firewall da organização, sendo esse um dos mecanismos de segurança que implementa os aspectos mais importantes da política, relacionados aos controles sistêmicos ou tecnológicos necessários à manutenção da segurança da informação.

O próprio firewall também precisa de uma política, a qual deve prever a necessidade de novas regras de filtragem. Sendo assim, a política do firewall pode simplesmente proibir a criação de novas regras, ou então exigir uma análise de segurança sobre o serviço que está sendo solicitado à liberação.

Pode-se dizer que uma política de segurança bem definida deixa o ambiente menos complexo do ponto de vista técnico, auxiliando a implementação e configuração de controles sistêmicos, como o firewall, provendo base e apoio ao administrador de segurança na tomada de decisões junto à alta direção.



Paulo Moreira
paulomoreira.ti@gmail.com
http://about.me/paulomoreirati
Profissional graduado em Sistemas de Informação, pós-graduado em Gestão de Segurança da Informação, certificado Microsoft e ITIL. Vivência na área de TI desde 2000, grande experiência na área de suporte, infraestrutura e em projetos da área, com foco em tecnologias Microsoft.

Referências

Firewalls Security Corporation
http://www.firewalls.com.br

ISACA Brasil
http://www.isaca.org.br

Módulo Security
http://www.modulo.com.br

FONTES, Edison. Praticando a Segurança da Informação. São Paulo: Brasport, 2008.

SÊMOLA, Marcos. Gestão da Segurança da Informação: Uma visão executiva. 9. ed. Rio de Janeiro: Elsevier, 2007.

Seu **conhecimento** **vale muito** na DevMedia

Publique artigos e vídeos gratuitamente no site e participe do programa **PayContent.**

Sua participação pode ser premiada: quanto mais você postar, mais chances de ganhar.



Para saber mais acesse:
www.devmedia.com.br/paycontent

DevMedia