Solis Pedro Eduardo da Silva

Utilização da Autenticação do *Samba* para Controlar o Acesso de Usuários à *Internet*

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Orientador
Professor Herlon Ayres Camargo

Lavras
Minas Gerais - Brasil
2006

Solis Pedro Eduardo da Silva

Utilização da Autenticação do *Samba* para Controlar o Acesso de Usuários à *Internet*

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Aprovada em *29 de setembro de 2006*

Professor Sandro Melo

Professora Simone Markenson Pech

Professor Herlon Ayres Camargo
(Orientador)

Lavras
Minas Gerais - Brasil

Agradecimentos

Agradeço especialmente aos meus Pais, Rosalina e Joaquim, à minha Esposa, Regina, aos meus filhos, Samuel Pedro Solis e João Pedro Solis, ao meu Orientador, Herlon, e aos demais professores do curso.

Resumo

Este trabalho apresenta uma proposta para alterações na solução para controle de acesso à *Internet Smbgate*. As alterações propostas envolvem melhorias no processo de controle do acesso à *Internet* dos usuários registrados no servidor *Samba* e gravação das atividades de cada usuário.

Dedico este trabalho aos meus Pais, Rosalina e Joaquim, à minha Esposa, Regina, e aos meus filhos, Samuel Pedro Solis e João Pedro Solis. Espero que um dia eu consiga lhes dar uma justa compensação por minhas ausências em seguidos finais de semana, noites e madrugadas adentro, sacrifício que fui obrigado a lhes impor, não sem remorso, para conseguir terminar este Curso.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Quando se fala em conceder acesso à *Internet* aos trabalhadores de uma empresa, a primeira preocupação é com relação à eventuais abusos que eles possam cometer. Ao invés da utilização da *Internet* como instrumento de trabalho ou para obtenção de informações necessárias à execução de tarefas, pode acontecer que a mesma seja usada para fins outros, como navegar por *sites* pornográficos, baixar fotos, músicas, vídeos e programas (inclusive "piratas"). Esta situação coloca em risco o sistema e as informações da empresa, pois aumenta a possibilidade de que, por exemplo, os computadores sejam infectados por um novo vírus ou cavalo de tróia. Representa também perda financeira e aumento de custos, de vez que o empregado perde tempo e ocupa recursos, o computador e largura de banda de conexão, para navegar por lugares que nada tem a ver com sua função dentro da empresa.

Torna-se então necessário controlar de alguma forma este acesso concedido à *Internet*. Este trabalho indica uma alternativa para implementação deste controle, através de adaptações e modificações em uma solução já existente, o *Smbgate* [1], um conjunto de *scripts* criado por Ricardo Alexandre Mattar. O *Smbgate* vale-se de uma máquina com o *Samba*, como controlador de domínio primário, e do filtro de pacotes *Iptables*, como um *proxy* transparente, para liberar ou bloquear o acesso do usuário à *Internet*, quando do registro do usuário no servidor *Samba*.

No Capítulo 2, inicialmente mostra-se definições sobre os sistemas *proxy* e como são utilizados para controlar o acesso à *Internet*. São mostrados os tipos de sistema *proxy* que podem ser implementados com o popular *software Squid*[2] e as vantagens e limitações deste sistema como um sistema *proxy* transparente. Também são mostradas algumas características de recursos existentes no *kernel*

[1] `http://www.tldp.org/HOWTO/Samba-Authenticated-Gateway-HOWTO.html`
[2] `http://www.squid-cache.org/`

do *Linux* e oferecidos pelo filtro de pacotes *Iptables*[3], os quais também podem ser usados para implementar o conceito de *proxy* transparente. No Capítulo 3, passa-se a analisar a solução já existente, o *Smbgate*, em sua implementação original. Algumas deficiências e recursos ausentes na solução, como por exemplo, procedimentos para registrar dados sobre os acessos à *Internet* utilizados e problemas com relação ao encerramento do acesso concedido, são discutidas.

As alterações no *Smbgate* são propostas no Capítulo 4. Além de resolver o problema do encerramento do acesso à *Internet* em uma máquina quando o usuário não registrou sua saída no servidor *Samba*, as mudanças indicadas neste Capítulo adicionam uma nova funcionalidade, o registro em arquivo dos acessos concedidos e das atividades dos usuários. Ao longo do texto pode-se ver que a remodelagem efetuada no *Smbgate* suprime deficiências existentes e oferece novos recursos, expandindo as possibilidades para sua utilização. As alterações ali sugeridas tornam possível que, em determinadas situações, sua nova versão seja usada em substituição a um sistema *proxy* tradicional, como o *Squid*, com vantagens, pois oferece autenticação de usuários e *proxy* transparente. Os resultados apresentados no texto comprovam esta potencialidade. Também são indicadas novas possibilidades de expansão deste trabalho.

O Capítulo 5 enfatiza o resultado positivo obtido com as modificações efetuadas no funcionamento do *smbgate*. Também são apresentadas considerações a respeito do controle de acesso à *Internet*, em relação aos sistemas operacionais em geral e em relação ao *Linux*, acentuando as vantagens deste, bem como sobre cuidados que devem ser tomados para sua implementação. São descritas em detalhes as alterações implementadas, no Apêndice A. No Apêndice B é transcrito um exemplo sobre a execução remota de comandos para alteração das regras de acesso, quando o servidor *Samba* fica em uma máquina e o *gateway* para acesso à *Internet* em outra máquina, finalizando então o trabalho.

A partir deste trabalho implementa-se a ferramenta *Smbgate* com as modificações pretendidas, entendendo-se plenamente como ela trabalha e utilizando-se da mesma em funcionamento para efetuar o controle de acesso à *Internet*. Mas além, o entendimento da ferramenta mostra que ela pode ser personalizada como solução para as mais diversas necessidades específicas, em relação àquele controle. Tudo isso usando a riqueza de recursos oferecidos pelo *Linux* e pelo *software* livre, sem qualquer custo, com segurança, estabilidade, desempenho, liberdade e transparência de funcionamento que estes ofereçem. Enfim, este trabalho é mais um exemplo do porquê o *Linux* e o *software* livre são considerados como ótimas opções para criação de soluções profissionais dentro do mundo da computação, fato confirmado cada dia mais pelo próprio mercado.

[3] `http://www.netfilter.org/projects/iptables/index.html`

Capítulo 2

Visão Geral do Controle de Acesso à *Internet*

O controle de acesso à *Internet*, na prática, é geralmente concretizado pela implantação de sistemas para filtragem de pacotes e serviços de *proxy*. Estes serviços são usualmente instalados, de forma integrada, em uma máquina conectada diretamente à *Internet*. Esta máquina torna-se então a responsável por prover acesso à *Internet* às demais máquinas da rede interna, sendo normalmente referenciada como *gateway*. A utilização de *gateways* facilita a gerência do acesso à *Internet* e permite maximizar o aproveitamento da conexão disponível. Também contribui para incrementar a segurança da rede interna contra ataques oriundos da *Internet*, uma vez que é possível concentrar sobre os *gateways* os esforços de prevenção e defesa, em diversos níveis.

Nos sistemas *Linux*, os aplicativos mais utilizados para filtragem de pacotes ou conteúdo e serviços de *proxy* é a dupla *Iptables* e *Squid*. Para a implementação de um sistema de controle de acesso naqueles sistemas, alguns conceitos sobre os aplicativos citados devem ser entendidos. Dentre eles, destacam-se os seguintes, dos quais trata este Capítulo:

- O que diferencia o funcionamento do *Iptables* e do *Squid*, uma vez que ambos podem ser usados como sistema *proxy*;
- Que recursos eles oferecem ou não, para implementação do controle de acesso à *Internet*;
- De que forma eles podem ser usados para esta tarefa.

2.1 Filtro de pacotes *Iptables*

"O Iptables *é um filtro de pacotes IP e estados"*(FILHO, 2005). Filtros IP operam basicamente na camada 2 da pilha de protocolos TCP/IP, ou seja, por definição somente são capazes de filtrar pacotes com base em informações do cabeçalho IP (endereço de origem e destino, TOS, TTL, protocolo, etc.). Mas o *Iptables* não se atém a esta definição. Conforme Andreasson (2005), possui recursos que permitem que trabalhe também na camada 3 da pilha de protocolos TCP/IP, com informações existentes nos cabeçalhos de protocolos como o UDP ou TCP, em camada inferior, com endereços MAC e também em relação ao estado das conexões.

O *Iptables*, porém, não considera protocolos acima do nível da camada de transporte da pilha de protocolos TCP/IP e nenhuma decisão baseada no conteúdo dos pacotes será tomada. Isso ocorre devido ao fato de que o *Iptables* é incapaz de conectar dados a partir de diferentes pacotes entre si. Desta maneira, não é próprio para tentar bloquear um fluxo de pacotes baseado numa determinada palavra, para evitar, por exemplo, um vírus ou o acesso a um determinado *site* no qual existam palavras "condenadas".

Além de filtrar pacotes, o *Iptables* oferece serviços de NAT (*Network Address Translation*), usados para alterar os endereços de origem e/ou destino dos pacotes, os denominado SNAT e DNAT. Estes serviços dependem da capacidade oferecida pelo *Linux* para rastreamento das conexões e dos pacotes, verificando se pertencem a um mesmo fluxo de dados. Esta capacidade é denominada *"connection tracking"*(ANDREASSON, 2005).

Quando o pacote entra na máquina com o filtro de pacotes *Iptables* em execução, ele percorre uma série de passos, antes de chegar à aplicação ou ser reencaminhado para outra máquina. Estes passos são chamados de *chains* e são mostrados na Figura 2.1. Em cada uma destas etapas poderão ser definidas regras com ações a serem tomadas com relação ao pacote, com base em suas características, em comparação (*match*) à valores pré-definidos em cada regra. Estas ações são classificadas em tabelas, de acordo com a finalidade da ação: *mangle*, *nat*, *filter* (a tabela padrão). Cada *chain* possui uma ação padrão, que será tomada a menos que uma regra específica para determinada *chain*, segundo sua tabela de classificação, indique o contrário. Dentre as principais ações, pode-se citar DROP, ACCEPT, REDIRECT, REJECT, LOG, DNAT, SNAT, MASQUERADE. Destaca-se neste trabalho, com base em seus objetivos, a *chain* FORWARD, na tabela *filter*, e a *chain* POSTROUTING, na tabela *nat*, com as ações DROP, ACCEPT, LOG, ULOG, SNAT e MASQUERADE.

Pela *chain* FORWARD, conforme se observa na Figura 2.1, passam os pacotes que serão reencaminhados pelo *kernel* do *Linux* para outra rede, no caso em ques-

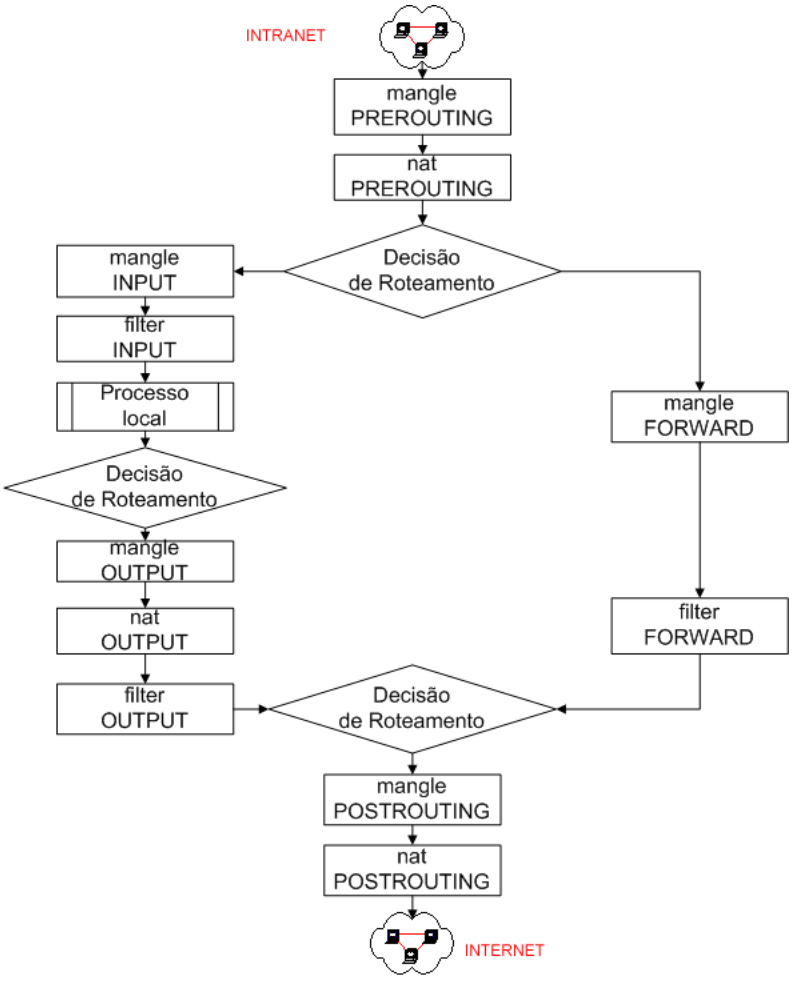

Figura 2.1: Encaminhamento de pacotes através das tabelas e *chains* do *Iptables* (ANDREASSON, 2005)

tão neste trabalho, a *Internet*. As regras desta *chain* na tabela *filter* determinam a quais pacotes será permitido o roteamento (ACCEPT) e quais serão eliminados (DROP). Esta escolha pode ser feita com base no protocolo e porta usados, endereço IP de origem ou destino do pacote, dentre várias outras opções.

A última *chain* que um pacote passa ao sair da máquina rumo à rede é a POSTROUTING. As regras da tabela *nat* definem como será alterado o endereço IP, usado na rede interna, para um endereço válido na *Internet*, com a ação SNAT ou MASQUERADE. Esta última é usada para alterar o endereço de origem do pacote para um endereço dinâmico configurado na *interface* de rede da *Internet*, que é

consultada a cada início de conexão. Já o SNAT é usado para definir um IP fixo para alteração do endereço de origem do pacote. Antes que a alteração do endereço de origem seja levada a termo, um registro de log pode ser feito, usando as ações LOG ou ULOG. A diferença entre uma e outra é que a segunda opção permite gravar o *log* em uma tabela de banco de dados *MySql*[1]. É importante lembrar que as informações que podem ser registradas no *log* são aquelas existentes nos cabeçalhos da camada 2 ou 3 da pilha de protocolos TCP/IP, ou seja, não é possível gravar no *log* o usuário que está acessando determinado destino, apenas o IP de origem e o IP de destino, por exemplo.

Embora exista a comparação (*match*) *owner*, que permite, apenas na *chain* OUTPUT, filtrar pacotes ou gravar informações sobre os pacotes no *log* com base na identidade do processo que gerou o pacote ou no usuário que iniciou o mesmo, ela não é muito eficaz. Isso ocorre, em primeiro lugar, porque existem pacotes que não possuem um dono, por exemplo, ICMP de resposta. Além disso, em segundo lugar, se o processo estiver rodando como outro usuário (*root*, por exemplo), os *logs* não registrarão o real usuário que estava usando a máquina no momento. Finalmente, apenas os pacotes de saída podem ser testados, os procedimentos de *log* deveriam ser efetuados em cada máquina, e não de forma centralizada no *gateway* de acesso à *Internet*. Esta última característica torna muito difícil a manutenção e análise dos registros individuais que seriam gerados em cada máquina.

Quando o *Iptables* faz SNAT, efetivamente se está diante de um serviço de *proxy* transparente, trabalhando na camada 2 da pilha de protocolos TCP/IP, pois endereços IP da rede interna estão sendo alterados para alcançar a rede externa. De forma inversa, uma vez estabelecida a conexão com a *Internet*, o mecanismo de *connection tracking* tornará possível que os pacotes que retornarem da rede externa recebam os endereços das respectivas máquinas às quais se destinam. *"O NAT é, na verdade, um* proxy *fundamental: um único host faz as solicitações em nome de todos os hosts internos, ocultando assim suas identidades da rede pública."*(STREBE; PERKINS, 2002)

2.2 Sistema *cache* e *proxy Squid*

"O Squid *é, ao mesmo tempo, um servidor proxy e de caching que roda sob o Linux e suporta diversos protocolos, entre os quais HTTP, FTP e SSL"*(NEMETH; SNYDER; HEIN, 2004). Embora muitas vezes seja tomado como sinônimo de "*proxy*", isto não é absolutamente correto. *"Um proxy é um software que atua como ponto entre duas redes, controlando o tráfego de acordo com seu conteúdo"*(UCHôA,

[1] `http://www.mysql.com/`

2005), ou ainda, um *proxy* é qualquer serviço que atue como intermediário (procurador é um termo geralmente utilizado como tradução para "*proxy*") entre um cliente e o servidor real dos serviços. Desta forma, teoricamente podem existir serviços de *proxy* para as diversas camadas do modelo TCP/IP, como de fato o próprio *Iptables*, que atua como um *proxy* quando provê serviços de SNAT/DNAT, conforme consta na Seção 2.1.

O *Squid* também oferece o serviço de *cache*. No caso do serviço HTTP, por exemplo, quando uma estação configurada como cliente de um servidor *proxy* Web fizer um pedido para baixar uma certa página, o navegador fará este pedido ao servidor *proxy Web*. Este servidor terá armazenadas as páginas visitadas recentemente e consultará este *cache* de páginas para ver se a página Web solicitada está disponível. Se estiver disponível, esta página armazenada no cache será enviada ao cliente. Se a página não estiver no cache, ou se a versão existente for muito antiga, o servidor *proxy* buscará a mesma do *site* em questão, armazenará essa página no seu cache e a enviará à estação de trabalho. *"Este mecanismo aumenta o desempenho da rede, porque a página da* Web *é baixada imediatamente para o cliente a partir do servidor* proxy*, evitando ter de baixá-la da Internet"*(STANGER; LANE, 2002).

O *Squid* atua como sistema *proxy* na camada de aplicativo do modelo TCP/IP, para os serviços HTTP e FTP. *"Para que trabalhe efetivamente como um intermediário entre a rede interna e a* Internet*, tem de ser usado em conjunto com algum método para restringir o tráfego no nível IP entre os clientes e os servidores reais, seja usando a filtragem de pacotes ou inibindo o roteamento de pacotes, pois se existir conectividade IP entre clientes e servidores reais, para os serviços do* proxy*, os clientes sempre poderão contornar o sistema* proxy*"*(ZWICKY; COOPER; CHAPMAN, 2000).

Com relação à sua utilização, existem duas formas básicas de configuração: como *proxy* "normal" ou como *proxy* "transparente". Da maneira normal, o serviço responde em uma porta específica no servidor (o padrão é a porta 3128) e os aplicativos nas estações de trabalho devem ser configurados para explicitamente acessar os serviços HTTP e FTP através desta porta. A principal desvantagem é que cada aplicativo tem de ser configurado desta forma para usar o *proxy*. Quando usado desta maneira, o *Squid* pode ser configurado para exigir autenticação do usuário, antes de atender suas requisições, sendo um meio efetivo de controlar quem pode e o que pode ser acessado na *Internet*.

A forma mais prática seria utilizar o *Squid* configurado como *proxy* transparente. Desta forma, o acesso à serviços específicos na *Internet* passa através do *proxy*, sem necessidade de se efetuar qualquer alteração nos aplicativos do usuário. Uma regra no filtro de pacotes *Iptables* redireciona todas as requisições feitas

pelos clientes, em uma das portas dos serviços providos pelo sistema *proxy*, para a porta em que o mesmo está respondendo pelo serviço, no servidor. Tais requisições são atendidas pelo *proxy* sem que isso se torne visível para o usuário. Assim, as requisições dos clientes sempre serão direcionadas ao servidor *proxy*, resolvendo o problema anteriormente citado, quando os clientes conseguem contornar o sistema *proxy* e acessar diretamente o serviço na *Internet*.

Segundo ViSolve Open Source Solutions (2006), a grande desvantagem desta forma de utilização é que não é possível ativar o recurso de autenticação de usuários, sem o que fica totalmente comprometido o controle de acesso e utilização. Os *logs* registrarão apenas as máquinas de origem das requisições e o que foi requisitado, sem indicar o usuário que estava usando efetivamente a máquina de origem do pedido em cada requisição. Esta desvantagem é a maior motivadora da realização deste trabalho, e de outros já existentes, como por exemplo o *Smbgate* que será discutido no Capítulo 3.

Uma grande vantagem dos sistemas *proxy* de serviços, como o *Squid*, é que eles permitem ao administrador acrescentar regras para filtrar conteúdo destes serviços, podendo serem bloqueadas palavras como sexo, pornografia, piadas e suas variantes. Também podem ser bloqueados *sites* específicos, de acordo com seu endereço ou domínio. Todas as requisições são registradas em *logs* e existem ferramentas para analisar estes arquivos de *log*. Através da análise do arquivo de *log* `acess.log` do *Squid*, em sistema configurado para exigir autenticação, pode-se obter diversas informações, tais como:

- Quais usuários acessaram quais *sites*, em que horas;
- quantos *bytes* foram baixados;
- quantas conexões foram feitas;
- relatórios de *sites* mais acessados;
- usuários que mais acessam;
- relatório de *sites* negados;
- falha de autenticação.

O SARG (*Squid Analysis Report Generator*)[2] é uma destas ferramentas que, conforme definição existente em seu *site*, *"permite ver onde seus usuários estão indo na* Internet*"*. Dois tutoriais para utilização do SARG podem ser consultados em `http://www.devin.com.br/eitch/sarg/` e em `http://www.zago.`

[2] `http://sarg.sourceforge.net/`

eti.br/squid/sarg.html, e mostram o grande potencial da ferramenta para análise do arquivo de *log* do *Squid*.

2.3 Diferenças entre um filtro de pacotes e um servidor *proxy*

"É comum a confusão entre as definições de um filtro de pacotes e um servidor proxy. *Ambos os serviços são colocados no perímetro da rede. Ambos funcionam como um intermediário entre uma LAN e a* Internet. *Ambos possuem a capacidade de filtrar o tráfego transmitido para dentro e para fora da rede. Ambos utilizam regras para determinar se certos tipos de tráfego terão autorização para passar pelo servidor ou se serão descartados"* (STANGER; LANE, 2002).

A diferença entre os dois, tomando por base o *Iptables* como filtro de pacotes e o *Squid* como servidor *proxy* de aplicação, é que o primeiro não penetra tão fundo no pacote como o *Squid*, em analogia a Stanger e Lane (2002). O *Iptables* usa informações das camadas 2 e 3 do modelo TCP/IP, podendo trabalhar com diversos protocolos. Já o *Squid* é um *proxy* da camada de aplicação, o que lhe oferece uma flexibilidade muito maior com relação a analisar o tráfego dentro de um serviço, como o tráfego da porta 80 (HTTP), determinando se deve ou não passar.

Em resumo, apenas com o *Squid* o administrador pode controlar conteúdo que está sendo acessado, nos serviços em que este atua como servidor *proxy*, basicamente HTTP/HTTPS e FTP. Já com o *Iptables* não é possível controlar o conteúdo, mas ele oferece diversas opções de controle dos pacotes individuais, com base nas informações dos cabeçalhos IP e da camada de transporte. Ademais, o *Iptables* atua ele próprio como um *proxy* da camada IP, quando executa o serviço de SNAT para que as máquinas da rede interna consigam acessar a *Internet*.

2.4 Considerações finais

A filtragem de pacotes é uma das maneiras de criar um perímetro de rede, torna possível controlar o que entra e sai da rede, com base na origem, destino ou estado de cada pacote, mas não com relação a seu conteúdo. O serviço de cache e *proxy* possibilita controlar o tráfego de forma bem mais específica, com base nos dados que estão sendo enviados ou recebidos. Também permite registrar o tráfego, de forma a identificar a máquina e/ou usuário responsável pelo mesmo, desde que configurado para fazer autenticação dos usuários. Entretanto, para usar autenticação de usuários é exigido um esforço maior para configurar cada máquina cliente.

Além disso, é possível aos usuários valerem-se de alguns artifícios para escapar da utilização do *proxy*, caso não sejam tomados cuidados com relação ao estabelecimento de restrições para as conexões IP. O *proxy Squid* configurado de forma a trabalhar de maneira "transparente" resolve esta situação, porém não trabalha com autenticação de usuário, conforme ViSolve Open Source Solutions (2006).

No próximo capítulo é analisado o *Smbgate*, uma solução para resolver este problema usando recursos do *Samba*, configurado como controlador de domínio primário, e do *Iptables*, como *proxy* transparente e filtro de pacotes.

Capítulo 3

Smbgate

Conforme o Capítulo 2, o controle de acesso à *Internet* pode ser feito pelo *Iptables*, pelo *Squid*, ou por ambos de forma integrada, cada qual atuando em diferentes graus de profundidade no modelo de camadas TCP/IP. Um controle efetivo do acesso é possível com a utilização da autenticação de usuário oferecida pelo *Squid*. Sua utilização como *proxy* transparente melhora o controle do acesso à *Internet*, pois torna obrigatória a utilização do *proxy* pelos usuários sem exigir alterações nas configurações usadas pelos mesmos em seus aplicativos. Entretanto, o *Squid* não aceita a utilização transparente como cache e *proxy* juntamente com a autenticação de usuários, segundo ViSolve Open Source Solutions (2006).

Neste Capítulo será descrita uma solução alternativa para criação de um sistema *proxy* transparente com autenticação de usuários. O *Smbgate*, criado por Ricardo Alexandre Mattar (MATTAR, 2005), utiliza para tal recursos do *Samba*, para autenticação dos usuários, aliado ao *Iptables*, atuando como filtro de pacotes e *proxy* transparente. O *Smbgate* original é bastante copiado, sem profundas alterações (infelizmente, na maioria das vezes, sem os devidos créditos a seu autor), em diversos *sites* pela *Internet*. É indicado como uma boa solução para resolver o problema da conjugação da utilização de *proxy* transparente e autenticação de usuário para controlar o acesso à *Internet*. Além disso, consta na lista de outras documentações, indicado no próprio *site* oficial do *Samba* (`http://us1.samba.org/samba/docs/`).

3.1 Recursos oferecidos pelo *Samba*

O principal recurso oferecido pelo *Samba* para tornar possível a implementação do *Smbgate* é a autenticação de usuários, por seu intermédio, como um controlador de domínio primário (PDC). Desta forma, apenas os usuários autenticados no *Samba*

estão habilitados a acessar a *Internet*. A solução é voltada para uma rede formada por máquinas *Windows*[1] que acessam a *Internet* através de um *gateway Linux* e que tem seus usuários autenticados em um servidor *Samba*.

O *Samba* oferece suporte para execução de *scripts* de logon *Windows*, arquivos .BAT ou .CMD, que serão executados no cliente quando um usuário se registra no domínio Windows (no caso, o servidor PDC *Samba*). Estes *scripts* de *logon*, que ficam armazenados no servidor *Linux*, são enviados ao cliente quando um usuário se registra no domínio e executados. *"Eles podem ser personalizados de acordo com o sistema operacional que acessa o PDC, por usuário ou simplesmente utilizar um* script *padrão para qualquer sistema ou usuário"* (MATTAR, 2005). O *script* de *logon* fica gravado dentro do diretório do compartilhamento `netlogon`, definido no arquivo de configuração `smb.conf` do servidor Samba.

Depois que o usuário completa a autenticação no servidor PDC *Samba*, o *script* de *logon* é executado, e este *script* pode determinar a montagem, no cliente, de um diretório compartilhado no servidor. Na definição do compartilhamento pode-se especificar um programa ou *script* a ser executado, no servidor, antes que ocorra a montagem do diretório compartilhado. Da mesma forma, existe uma opção para especificar um programa ou *script* a ser executado, também no servidor, depois que a conexão a um compartilhamento é finalizada (o diretório compartilhado é desmontado). A Tabela 3.1 mostra um resumo destas opções.

Tabela 3.1: Opções para indicação de *scripts* de conexão
Fonte: Eckstein, Collier-Brown e Kelly (1999)

Opção	**Paramêtros**	**Função**
root preexec	texto (comando *Linux*)	Indica um comando para executar como root, antes de conectar a um compartilhamento.
preexec	texto (comando *Linux*)	Indica um comando para executar como usuário comum antes de conectar a um compartilhamento.
postexec	texto (comando *Linux*)	Indica um comando para executar como usuário comum após desconectar um compartilhamento.
root postexec	texto (comando *Linux*)	Indica um comando para executar como root, depois de desconetar um compartilhamento.

[1] Nada impede que o acesso à *Internet* de máquinas *Linux* também possa controlado usando o *Smbgate*, conforme considerações na Seção 3.3.

3.2 O *Smbgate* em funcionamento

O roteiro para implementação do *Smbgate* original, pode ser consultado em Mattar (2005), e com base na implementação realizada a partir desta fonte, faz-se nesta seção uma análise de seu funcionamento. As listagens de arquivos de *script* estão no seu formato original.

Com as estações Windows configuradas como membros do domínio gerenciado pelo PDC *Samba*, ao se registrar no domínio elas receberão e executarão o *script* de *logon* padronizado. Ele deve estar no compartilhamento `netlogon` e é especificado através da linha `logon script = NETLOGON.BAT` existente no arquivo de configuração do *Samba* `smb.conf`. Este *script*, mostrado na Figura 3.1, determinará a montagem, no cliente, de um diretório compartilhado no servidor. Este diretório compartilhado no servidor será usado para rastrear o usuário e seu endereço IP, conforme adiante descrito.

```
1  REM NETLOGON.BAT
2  net use z: \\linux\samba /yes
```

Figura 3.1: *Script* de *logon* NETLOGON.BAT que será executado na máquina cliente (MATTAR, 2005)

Conforme mostrado na Figura 3.2, na definição do compartilhamento usado para rastreamento existem duas opções: `root preexec` e `root postexec`. Elas indicam *scripts Linux* que serão executados no servidor, antes de cada montagem e depois de cada desmontagem do diretório compartilhado, e são explicados na Tabela 3.1. Estes *scripts* recebem como parâmetro o nome e endereço IP do usuário e neles reside a lógica principal do funcionamento. São eles que acionam um terceiro *script*, para efetivamente alterar as regras do *Iptables*, liberando ou bloqueando o acesso à *Internet*.

```
1  [samba]
2  comment = login tracking share
3  path = /home/samba/samba
4  browseable = No
5  root preexec = /usr/local/bin/netlogon.sh %u %I
6  root postexec = /usr/local/bin/netlogoff.sh %u
```

Figura 3.2: Compartilhamento, usado para rastreamento dos dados do usuário e IP de sua máquina (MATTAR, 2005)

O *script* `netlogon.sh`, mostrado na Figura 3.3, é iniciado quando ocorre a montagem do diretório compartilhado usado para rastreamento, conforme definido na Figura 3.2. Sua principal função é selecionar outro *script* a ser executado, com base no nome do usuário ou com base no grupo ao qual ele pertence. Em máquinas Windows, a montagem do diretório compartilhado usado para rastreamento, que dispara a execução do `netlogon.sh`, ocorre em função da execução do *script* de *logon* NETLOGON.BAT. Este último é mostrado na Figura 3.1 e sua execução ocorre no momento que o usuário faz seu registro no PDC *Samba*, sendo justamente este o motivo de ser chamado de *script* de *logon*.

O *script* `netlogon.sh` grava o IP da máquina do usuário em um arquivo em `/var/run/smbgate`, com o objetivo de usar este dado para rastreamento. Este arquivo terá o nome do usuário e será utilizado mais tarde quando o usuário desconectar. Em seguida, é armazenado o grupo do usuário e o endereço IP da máquina do usuário é repassado como argumento para o outro *script* que será responsável pela atualização das regras do *Iptables*. Nota-se que primeiramente é testada a existência de um *script* com o nome do usuário. Se este não for localizado, parte-se para a busca de um *script* com o nome do grupo do usuário e, por último, se este também não for localizado, um *script* padrão. Este comportamento está ilustrado na Figura 3.4.

De forma inversa, o *script* `netlogoff.sh`, listado na Figura 3.5, é executado quando o compartilhamento de rastreamento é desmontado, ou seja, quando é registrada a saída do usuário no PDC *Samba*. Esta definição pode ser vista na Figura 3.2. Após confirmar que o usuário ainda está conectado ao PDC, ele repassa o endereço IP extraído do arquivo gravado com seu nome em `/var/run/smbgate/` como argumento para o *script* responsável pela atualização das regras do *Iptables*. Também neste caso, primeiramente é tentado um *script* com o nome do usuário, se não for localizado, parte-se para a busca de um *script* com o nome do grupo do usuário e, por último, se este também não for localizado, um *script* padrão. Este comportamento está ilustrado na Figura 3.6

O *script* responsável pela atualização das regras do *Iptables* pode ser definido para cada usuário, grupo ou um arquivo padrão. Na Figura 3.7 é mostrado este arquivo, que recebe como parâmetros o comando, indicando a ação a ser feita, incluir ou excluir uma regra, o endereço e a *interface* de rede, repassados pelos *scripts* `netlogon.sh` ou `netlogoff.sh`.

3.3 Considerações sobre o *Smbgate*

Da forma como proposto originalmente, o *Smbgate* irá funcionar adequadamente desde que todos os usuários que façam registro de entrada no PDC *Samba* também

```
1  #!/bin/sh
2  #
3  # netlogon.sh
4  #
5  # usage:
6  # netlogon.sh <username> <ip>
7  #
8  if [ -f /var/run/smbgate/$1 ] ; then
9      exit 0
10 fi
11 echo $2 > /var/run/smbgate/$1
12 IPTABLES='/usr/sbin/iptables'
13 EXTIF='eth0'
14 COMMAND='-A'
15 ADDRESS=`cat /var/run/smbgate/$1`
16 GROUP=`groups $1 | gawk '// { print $3 }'`
17 if [ -f /etc/smbgate/users/$1 ] ; then
18     /etc/smbgate/users/$1 $COMMAND $ADDRESS $EXTIF
19 else
20     if [ -f /etc/smbgate/groups/$GROUP ] ; then
21      /etc/smbgate/groups/$GROUP $COMMAND $ADDRESS $EXTIF
22     else
23      /etc/smbgate/users/default.sh $COMMAND $ADDRESS $EXTIF
24     fi
25 fi
```

Figura 3.3: *Script netlogon.sh* acionado na montagem do compartilhamento para rastreamento (MATTAR, 2005)

registrem sua saída. Isso é uma situação ideal, pois na prática é comum acontecer o contrário, seja por travamento ou desligamento incorreto da máquina, seja por outros motivos.

Quando um usuário registra sua entrada no sistema, o *Smbgate* verifica se existe um arquivo com o nome deste usuário gravado em `/var/run/smbgate/`. Se existir, termina sem fazer mais nada, e esta será a causa de diversos problemas, pois irá provocar inconsistências nos registros existentes no PDC sobre o usuário e o IP de sua máquina em relação à situação real naquele momento. Este arquivo com o nome do usuário existirá, no momento de registro de nova entrada deste mesmo usuário, somente em duas situações: ou este usuário saiu do PDC sem registrar sua saída (travamento ou desligamento da máquina) ou está registrando novamente sua entrada, a partir de uma segunda máquina.

Qualquer que seja o motivo, a utilização da *Internet* não será autorizada neste novo registro de entrada do usuário, pois, como anteriormente citado, o *script* ter-

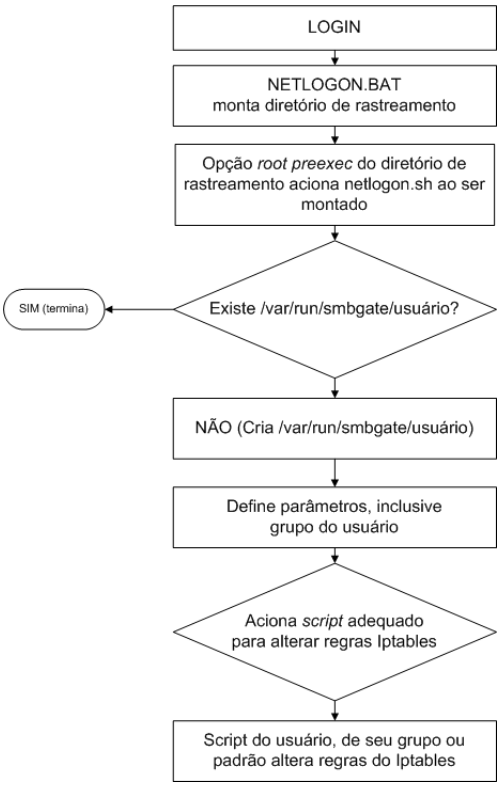

Figura 3.4: Diagrama de funcionamento do *script netlogon.sh*

mina imediatamente. Entretanto, se o este usuário estiver registrando sua entrada na mesma máquina na qual anteriormente já o havia feito, conseguirá acessar a *Internet*, pois a regra anteriormente inserida no *Iptables* não foi excluída. Reside aí um dos maiores problemas: qualquer outro usuário que registrar sua entrada no PDC nesta mesma máquina poderá usar a *Internet*. Isso até que o usuário que registrou sua entrada e não saiu faça nova entrada na mesma máquina e registre sua saída, normalizando a situação.

Para resolver este problema, pode-se usar a opção `keep alive`, indicada na *man page* do arquivo de configuração `smb.conf`, para forçar a verificação periódica do estado das conexões efetuadas pelos clientes. Esta opção define um intervalo de tempo, em segundos, para que periodicamente as conexões dos usuários registrados sejam verificadas. Se encontrada alguma conexão não ativa, será forçada a desconexão daquele usuário. Isso vai fazer com que ocorra a desmontagem dos compartilhamentos montados, inclusive daquele usado para rastreamento. Em

```
1  #!/bin/sh
2  #
3  # netlogoff.sh
4  #
5  # usage:
6  # netlogoff.sh <username>
7  #
8  IPTABLES='/usr/sbin/iptables'
9  EXTIF='ppp0'
10 COMMAND='-D'
11 TRACKSHARE="samba"
12 ADDRESS=`cat /var/run/smbgate/$1`
13 GROUP=`groups $1 | gawk '// { print $3 }'`
14 NM=`smbstatus -u $1 | grep $TRACKSHARE | wc -l`
15 if [ $NM -gt 0 ]; then
16     exit
17 fi
18 if [ -f /etc/smbgate/users/$1 ] ; then
19     /etc/smbgate/users/$1 $COMMAND $ADDRESS $EXTIF
20 else
21     if [ -f /etc/smbgate/groups/$GROUP ] ; then
22      /etc/smbgate/groups/$GROUP $COMMAND $ADDRESS $EXTIF
23     else
24      /etc/smbgate/users/default.sh $COMMAND $ADDRESS $EXTIF
25     fi
26 fi
27 rm -f /var/run/smbgate/$1
```

Figura 3.5: *Script netlogoff.sh* acionado na desmontagem do compartilhamento para rastreamento (MATTAR, 2005)

conseqüência, será executado o *script* indicado na opção `root postexec`, que vai alterar as regras no *Iptables*.

Em Mattar (2005) são propostas alternativas, um *script* para ser executado periodicamente, agendado no `cron`[2], ou uma outra implementação do *Smbgate*, através de um *script* colocado em `/etc/init.d/`. Ambas propostas, cada qual a sua maneira, tem em comum o fato de periodicamente verificarem os usuários registrados no servidor *Samba* e, a partir desta informação, ajustar as regras do *Iptables*.

Todos estes procedimentos efetivamente funcionam, mas em todos eles ocorre um lapso de tempo até que o fim da autorização ocorra, e no máximo será igual

[2]O cron é um recurso padrão do Unix (e do *Linux*) que permite o agendamento de tarefas para serem executadas em um momento específico, ou em intervalos regulares.

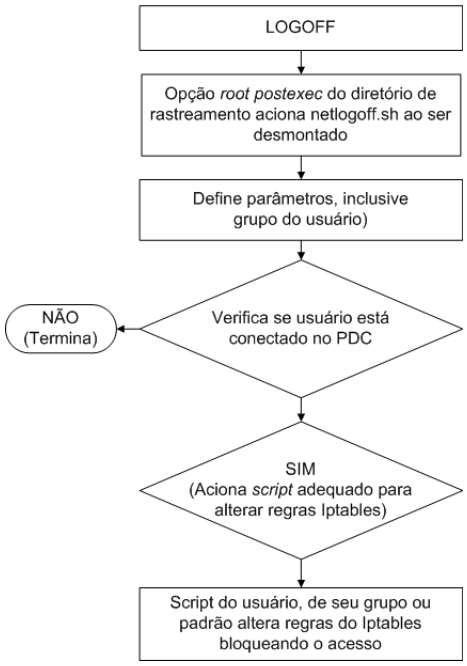

Figura 3.6: Diagrama de funcionamento do *script netlogoff.sh*

```
1  #!/bin/sh
2  #
3  COMMAND=$1
4  ADDRESS=$2
5  EXTIF=$3
6  IPTABLES='/usr/sbin/iptables'
7  $IPTABLES $COMMAND POSTROUTING -t nat -s $ADDRESS -o $EXTIF \
8  -j MASQUERADE
```

Figura 3.7: *Script* responsável pela alteração das regras no *Iptables* (MATTAR, 2005)

ao intervalo entre a execução das rotinas. Se for um período muito curto, a execução periódica das rotinas podem sobrecarregar o sistema e prejudicar o tempo de resposta. De toda forma, mesmo que por pouco tempo, usuários que não deveriam ter acesso à *Internet* poderão eventualmente conseguir fazê-lo, e isso pode ser inaceitável segundo as regras de controle impostas pela política de segurança em uso. Observa-se ainda, no caso da implementação usando *scripts* executados em períodos regulares, que a própria liberação do acesso não se dará imediatamente

após o registro do usuário no servidor *Samba*. Esta liberação somente vai ocorrer na próxima execução do *script*.

Uma observação final sobre esta solução diz respeito às máquinas *Linux* existentes na rede. A menos que sejam configuradas para autenticar seus usuários no PDC *Samba*, não teriam como executar o *script* de logon (que neste caso também teria de ser personalizado de acordo com cada sistema operacional existente nos clientes). Entretanto, os *scripts* determinados para executar antes da montagem e após a desmontagem do compartilhamento usado para rastreamento são acionados normalmente na sua montagem e desmontagem, como seria de se esperar, usando o comando `mount` com o tipo de sistema de arquivos `smbfs` no cliente *Linux*. Este fato indica que esta é uma possível saída para resolver esta questão.

3.4 Considerações finais

A solução oferecida pelo *Smbgate* permite a utilização de um *proxy* transparente com autenticação do usuário, usando recursos oferecidos pelo *Iptables* e pelo *Samba*. Entretanto, nota-se que em relação à questão do encerramento da autorização os resultados são incertos. Em certas circunstâncias, este encerramento do acesso do usuário à *Internet* pode não acontecer imediatamente, e isso pode não ser aceitável em relação aos quesitos de controle de acesso exigidos. No Capítulo 4 é modelada uma proposta para resolver pontualmente este problema e incorporar capacidade de criação de registros em arquivo das atividades do *Smbgate* e do próprio *proxy*.

Capítulo 4

Nova Proposta de Funcionamento para o *Smbgate*

O *Smbgate* foi desenvolvido em módulos, o que permite grande versatilidade em se adaptar o mesmo aos mais diversos propósitos, sem grande esforço para reescrever o código necessário para tal. Exatamente isso é feito neste Capítulo, onde é descrita uma nova proposta alterando o funcionamento do *Smbgate*. As alterações não somente buscam sanar as deficiências apontadas na análise efetuada no Capítulo anterior, como também incluir funcionalidade de registro das suas atividades em conjunto com as atividades do *Iptables*.

4.1 Alterações propostas

Como o objetivo é controlar e registrar o acesso de cada usuário, não haverá necessidade de utilização dos recursos para conceder acesso por grupo ou acesso padrão, portanto a parte do código que implementa esta funcionalidade será eliminada. Nenhuma alteração se fará necessária em relação ao *script* de *logon* bem como às configurações sugeridas para o *Iptables*, para o *Samba* e estrutura de diretórios.

A grande alteração irá ocorrer no arquivo `netlogon.sh`, na Figura 4.1 mostrado em sua nova versão. No momento do registro da entrada de um usuário, além de criar um arquivo com o nome do usuário contendo o IP de sua máquina, o *script* `netlogon.sh` passa a criar também um arquivo cujo nome será o IP da máquina e o conteúdo o nome do usuário que registrou a entrada a partir da mesma. Desta forma será possível fazer uma dupla verificação, se o usuário ou a máquina estão registrados como conectado no PDC. Se algum dos dois estiver nesta situação, pri-

meiramente será acionado o *script* `netlogoff.sh`, para restaurar a situação a um estado consistente, antes de prosseguir e incluir uma nova regra no *Iptables*.

```
1  #!/bin/sh
2  # netlogon.sh
3  # uso:
4  # netlogon.sh <username> <IP>
5  #Armazena dia e hora atual, para uso no registro
6  MOMENTO=`date +%Y%m%d%H%M%S`
7  # Se o IP foi registrado em logon anterior
8  if [ -f /var/run/smbgate/$2 ] ; then
9     USUARIO=`cat /var/run/smbgate/$2`
10    echo "FORCADO :" $2 $USUARIO >>/var/log/smbgate/smbgate.log
11    /usr/local/bin/netlogoff.sh $USUARIO
12 fi
13 # Se o usuario for autorizado (possuir script)
14 if [ -f /etc/smbgate/users/$1 ] ; then
15    # Se o usuario foi registrado em logon anterior
16    # Forca o logoff
17    if [ -f /var/run/smbgate/$1 ] ; then
18       echo "FORCADO :" $1 >> /var/log/smbgate/smbgate.log
19       /usr/local/bin/netlogoff.sh $1
20    fi
21    echo $2 > /var/run/smbgate/$1
22    echo $1 > /var/run/smbgate/$2
23    #log de inicio de liberacao de acesso aqui.
24    echo "LOGON :"$2 $MOMENTO $1 >>/var/log/smbgate/smbgate.log
25    #Preparação de parâmetros
26    EXTIF='eth0'
27    COMMAND='-A'
28    ADDRESS=`cat /var/run/smbgate/$1`
29    #Aciona o script para alterar as regras do Iptables
30    /etc/smbgate/users/$1 $COMMAND $ADDRESS $EXTIF
31 fi
```

Figura 4.1: Arquivo netlogon.sh alterado (com base em Mattar (2005))

Para registrar o momento que o acesso foi liberado para cada usuário e, posteriormente, este acesso foi bloqueado quando do registro de sua saída do PDC, é usado o arquivo `smbgate.log` que pode ser colocado em `/var/log/smbgate/`. Os registros ocorrem nos *scripts* `netlogon.sh` e `netlogoff.sh`. Neste último arquivo, mostrado na Figura 4.2, a inclusão de uma linha para fazer o registro é a principal mudança, juntamente com a inclusão da linha para exclusão do arquivo com o nome do IP da máquina.

```
1  #!/bin/sh
2  # netlogoff.sh
3  # uso:
4  # netlogoff.sh <username>
5  if [ -f /var/run/smbgate/$1 ]; then
6     EXTIF='eth0'
7     COMMAND='-D'
8     ADDRESS=`cat /var/run/smbgate/$1`
9     /etc/smbgate/users/$1 $COMMAND $ADDRESS $EXTIF 2> /dev/null
10    MOMENTO=`date +%Y%m%d%H%M%S`
11    echo "LOGOFF  :" $ADDRESS $MOMENTO $1 \
12       >> /var/log/smbgate/smbgate.log
13    rm -f /var/run/smbgate/$1 $ADDRESS 2> /dev/null
14 fi
```

Figura 4.2: Arquivo netlogoff.sh alterado (com base em Mattar (2005))

Finalmente, o *script* que altera as regras do *Iptables* foi alterado de forma que, além de liberar o NAT para os pacotes vindos do IP da máquina do usuário, seja registrado no arquivo de registro do *Iptables* todos os pacotes cujo estado seja "NEW", tendo como prefixo o nome do usuário. Esta última informação é obtida a partir do próprio nome do *script* (que é o nome de cada usuário que deve ter acesso à *Internet*). Estes registros são ajustados no nível de alerta, e o arquivo `/etc/syslog.conf` sofre um ajuste para direcionar o registro dos alertas do kernel para o arquivo `/var/log/iptables`, através da adição da seguinte linha ao mesmo: `*.alert /var/log/iptables`.

4.2 Resultados obtidos

As mudanças resolveram o problema da falta de encerramento da autorização de um usuário quando sua máquina trava ou é desligada incorretamente e sua saída não é registrada no PDC *Samba*.

Quanto ao registro das ações efetuadas de autorização de acesso e encerramento da mesma, o resultado pode ser observado na Figura 4.4, o qual indica, inclusive, ocorrências de desconexão forçada, ou seja, quando o usuário, por algum motivo, não registrou sua saída do PDC e efetuou novo login, na própria máquina na qual inicialmente havia registrado entrada ou em outra máquina. O "LOGOFF" registrado na segunda do arquivo não foi decorrente de uma saída normal do usuário solispedro. A máquina foi desligada, mesmo assim, aproximadamente vinte minutos depois ocorreu o registro da saída do usuário solispedro, o que acionou a desmontagem do volume de compartilhamento e conseqüentemente

```
1  !/bin/sh
2  #
3  USER=`echo $0 | cut -c20-`
4  COMMAND=$1
5  ADDRESS=$2
6  EXTIF=$3
7  IP="10.1.1.9"
8  IPTABLES='/sbin/iptables'
9  $IPTABLES $COMMAND POSTROUTING -t nat -s $ADDRESS -o $EXTIF \
10   -m state --state NEW \
11   -j LOG --log-level alert --log-prefix \
12   $USER" " --log-ip-options
13 $IPTABLES $COMMAND POSTROUTING -t nat -s $ADDRESS -o $EXTIF \
14   -j SNAT --to $IP
15 exit 0
```

Figura 4.3: Arquivo para alteração das regras do *Iptables* modificado (com base em Mattar (2005))

o *script* `netlogoff.sh`, que regularizou a situação e registrou o fato no arquivo. Na linha 3 o usuário solispedro entrou no sistema e também a máquina foi desligada. Quando fez novamente registro de entrada, a saída forçada foi registrada, conforme consta na linha 4, antes do registro da saída.

```
1  LOGON   : 192.168.0.2 20060720001119 solispedro
2  LOGOFF  : 192.168.0.2 20060720003521 solispedro
3  LOGON   : 192.168.0.2 20060721045647 solispedro
4  FORCADO : solispedro
5  LOGOFF  : 192.168.0.2 20060721050000 solispedro
6  LOGON   : 192.168.0.2 20060721050139 solispedro
7  LOGOFF  : 192.168.0.2 20060721052329 solispedro
8  LOGON   : 192.168.0.2 20060721082840 jpedro
9  LOGOFF  : 192.168.0.2 20060721082939 jpedro
10 LOGON   : 192.168.0.2 20060721083012 solispedro
11 LOGOFF  : 192.168.0.2 20060721083438 solispedro
```

Figura 4.4: Trecho do arquivo de registro gerado pelo *Smbgate*

Temos na Figura 4.5 o outro registro, agora gerado pelo *Iptables*, indicando a data e hora e as novas conexões iniciadas, com o nome do usuário, IP de origem, IP de destino e outras informações presentes no cabeçalho IP dos pacotes.

```
1  Jul 21 08:28:55 maqsmb kernel: jpedro IN= OUT=eth0
2  SRC=192.168.0.2 DST=64.233.179.99 LEN=48 TOS=0x00 PREC=0x00
3  TTL=127 ID=3411 DF PROTO=TCP SPT=2693 DPT=80 WINDOW=16384
4  RES=0x00 SYN URGP=0
5  Jul 21 08:30:27 maqsmb kernel: solispedro IN= OUT=eth0
6  SRC=192.168.0.2 DST=200.131.250.242 LEN=48 TOS=0x00 PREC=0x00
7  TTL=127 ID=8471 DF PROTO=TCP SPT=2708 DPT=443 WINDOW=16384
8  RES=0x00 SYN URGP=0
9  Jul 21 08:30:27 maqsmb kernel: solispedro IN= OUT=eth0
10 SRC=192.168.0.2 DST=209.73.168.74 LEN=48 TOS=0x00 PREC=0x00
11 TTL=127 ID=8483 DF PROTO=TCP SPT=2710 DPT=443 WINDOW=16384
12 RES=0x00 SYN URGP=0
```

Figura 4.5: Trecho do arquivo de registro gerado com *Iptables*

4.3 Considerações finais

Esta nova versão do *Smbgate* torna possível o registro das autorizações para acesso à *Internet* através do *proxy* transparente e o registro com o nome do usuário da origem e do destino do tráfego autorizado pelo *Iptables*. Os resultados obtidos mostram que é uma alternativa realmente eficaz à utilização do *Squid* com autenticação de usuários, cumprindo totalmente a meta de verificar por onde cada usuário navegou na *Internet*. E tudo isso, sem perder as vantagens de se utilizar um *proxy* de forma transparente para os usuários.

Como possibilidade de aprimoramento deste trabalho a sugestão é ajustar as regras que são alteradas no *Iptables*, para que o próprio *Squid* opere como um *proxy* transparente. O registro de autorizações de acesso gerado pelo *Smbgate* permite vincular o tráfego registrado pelo *Squid* a cada usuário, através da data, hora e IP de origem registrados, através de um *script*. Outra possibilidade é adequar os próprios registros feitos pelo *Iptabes* no *Smbgate* e criar uma ferramenta para gerar análises e relatórios a partir destes registros, disponibilizando informações de forma semelhante às disponibilizadas pelas ferramentas existentes para análise de registros do *Squid*. Finalmente, bem delineado neste trabalho o processo para utilização do *Smbgate* para liberar o acesso para máquinas *Windows*, há a necessidade de estabelecer, testar e documentar o processo necessário para sua utilização com máquinas *Linux*.

Observa-se que, nestas implementações, o PDC *Samba* está instalado junto com o *gateway* numa única máquina e os testes foram efetuados em um sistema configurado desta maneira. Certamente esta não é a situação geralmente encontrada em um ambiente real, onde o *Samba* provavelmente estaria numa máquina e o *gateway* em outra máquina. Esta questão não impede o uso do *Smbgate*: usando

o *SSH*[1], a partir da máquina *Samba*, pode-se executar remotamente o *Iptables* no *gateway*. No Apêndice B encontra-se a tradução livre, a partir de Mattar (2005), especificamente da parte daquela documentação que mostra um exemplo passo-a-passo desta utilização do *SSH*.

[1] `http://www.openssh.org/`

Capítulo 5

Conclusão

A partir da leitura deste documento pode-se não apenas compreender como implementar uma ferramenta relativamente simples, mas muito eficiente para cumprir o propósito a que se destina, qual seja controlar e registrar o acesso à *Internet* efetuado pelos usuários de uma rede de computadores. Outro conhecimento adjacente que será adquirido é a compreensão do poder do sistema operacional *Linux*, especialmente das linguagens de *scripts* que utiliza.

De forma diferente de outros sistemas operacionais, foi possível personalizar totalmente o controle de acesso à *Internet*, sem gastar um centavo sequer com programas específicos para isso, apenas utilizando a linguagem de *script* e recursos oferecidos por software livre. Desta forma, é impensável no mundo atual que um administrador de rede desconsidere totalmente o *Linux* como uma das melhores opções para implantação de servidores. Esta vantagem se dá justamente em função de características como as acima exemplificadas, além de muitas outras vantagens, como mais recursos na área de segurança e melhor desempenho, inclusive em determinados tipos de utilização, mesmo se usado *hardware* mais antigo.

Quanto ao controle de acesso à *Internet*, atualmente mostra-se como uma necessidade essencial, pois o tempo das pessoas é um recurso escasso e a quantia gasta para o pagamento das mesmas também o é. Diante da acirrada concorrência do mercado, as empresas devem buscar todas as formas para otimizar sua eficiência operacional, ou seja, produzir mais com menor custo. Certamente funcionários acessando a *Internet* de forma descontrolada em nada contribui para o alcançe deste objeto. Entretanto, especialmente quanto ao registro do que foi acessado, é importante lembrar que a informação de que isso poderá ser feito deve ser formalmente comunicada a todas as pessoas. Esta é uma precaução importante para resguardar-se contra possíveis reclamações de que tais registros ferem o direito individual à privacidade.

Tomados estes cuidados, a versão do *Smbgate* modificada representa uma boa opção para implementação deste controle, e os registros gerados podem ser úteis, seja como prova da prática de irregularidades ou como gerador de informações para análise do tipo de uso que está sendo feito da *Internet* colocada à disposição dos funcionários.

Referências Bibliográficas

ANDREASSON, O. *Iptables Tutorial 1.2.0*. [S.l.], 2005. Disponível em: <http://iptables-tutorial.frozentux.net/iptables-tutorial.html>.

ECKSTEIN, R.; COLLIER-BROWN, D.; KELLY, P. *Using Samba*. [S.l.], 1999. Disponível em: <http://doc.async.com.br/using_samba/index.html>.

FILHO, J. E. M. *Firewall com Iptables*. [S.l.], 2005. Disponível em: <http://www.eriberto.pro.br/iptables/>.

MATTAR, R. A. *Samba Authenticated Gateway HOWTO*. [S.l.], 2005. Disponível em: <http://www.tldp.org/HOWTO/Samba-Authenticated-Gateway-HOWTO-.html>.

NEMETH, E.; SNYDER, G.; HEIN, T. R. *Manual Completo do Linux*. São Paulo: Pearson Makron Books, 2004.

STANGER, J.; LANE, T. P. *Rede Segura Linux*. Rio de Janeiro: Alta Books, 2002.

STREBE, M.; PERKINS, C. *Firewalls*. São Paulo: Makron Books, 2002.

UCHôA, J. Q. *Segurança Computacional*. 2. ed. Lavras: UFLA/FAEPE, 2005. (Curso de Pós-Graduação "Lato Sensu" (Especialização) a Distância: Administração de Redes Linux).

VISOLVE OPEN SOURCE SOLUTIONS. *Squid Configuration Manual*. [S.l.], 2006. Disponível em: <http://squid.visolve.com/squid/squid24s1/access_controls-.htm>.

ZWICKY, D. E.; COOPER, S.; CHAPMAN, D. B. *Construindo Firewalls para a Internet*. Rio de Janeiro: Campus, 2000.

Apêndice A

Implantação do *Smbgate* com alterações

Neste apêndice estão os códigos de todos os arquivos de configuração utilizados, após uma indicação inicial da estrutura de diretórios usada. A rede usada para implantação é composta de dois computadores: o *gateway* com duas placas de rede, configuradas com os IPs 10.1.1.9 e outra com o IP 192.168.0.1. No *gateway* a rota padrão é 10.1.1.1, um modem ADSL configurado como roteador. Na outra máquina, com o IP 192.168.0.2, a rota padrão era 192.168.0.1.

```
1 /public
2 /etc/smbgate
3 /etc/smbgate/users
4 /home/netlogon
5 /home/profiles
6 /usr/local/bin
7 /var/run/smbgate
8 /var/log/
9 /var/log/smbgate
```

Figura A.1: Estrutura de diretórios utilizada (com base em Mattar (2005))

```
1  #!/bin/bash
2  INTRANET="192.168.0.0/24"
3  sysctl -w net.ipv4.ip_forward=1
4  iptables -F
5  iptables -t nat -F
6  iptables -t mangle -F
7  iptables -P INPUT DROP
8  iptables -P FORWARD DROP
9  iptables -P OUTPUT ACCEPT
10 iptables -A FORWARD -d $INTRANET -j ACCEPT
11 iptables -A FORWARD -s $INTRANET -j ACCEPT
12 iptables -A INPUT -m state --state ESTABLISHED,RELATED \
13   -j ACCEPT
14 iptables -A FORWARD -m state --state ESTABLISHED,RELATED \
15   -j ACCEPT
```

Figura A.2: Regras iniciais para *Iptables* (com base em Mattar (2005))

```
1 #/etc/syslog.conf
2 #Linha a ser incluida indicado o arquivo
3 #/var/log/iptables para gravação dos alertas
4 #(o log do tráfego do iptables foi ajustado
5 #para este nível
6 *.alert          /var/log/iptables
```

Figura A.3: Direcionamento de registros de alerta do *kernel*

```
1  workgroup =solistux.myn
2  server string = Servidor Samba TCC
3  log file = /var/log/samba/%m.log
4  max log size = 70
5  debug level = 1
6  local master = no
7  os level = 99
8  domain master = yes
9  preferred master = yes
10 domain logons = yes
11 wins support = yes
12 wins proxy = yes
13 logon home = \\%L\%u
14 logon script = netlogon.bat
15 logon path = \\%L\Profiles\%U
16 security = user
17 encrypt passwords = yes
18 smb passwd file = /etc/samba/smbpasswd
19 username map = /etc/samba/smbusers
20 socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
21 add machine script =/usr/sbin/adduser -n -r -g maqsamba \
22   -c "Grupo Maquinas Samba" -d /dev/null -s /bin/false %u
23 passdb backend = smbpasswd
24 ...
25 [publico]
26   path = /public
27     comment = pasta publica
28     writable = yes
29     share modes = no
30     guest ok = yes
31     force create mode = 777
32     force directory mode = 0777
33     browseable = yes
34     root preexec = /usr/local/bin/netlogon.sh %u %I
35     root postexec = /usr/local/bin/netlogoff.sh %u
```

Figura A.4: Principais opções usadas no arquivo *smb.conf* (com base em Mattar (2005)

```
#!/bin/sh
# /usr/local/bin/netlogon.sh
# uso:netlogon.sh <username> <IP>
MOMENTO=`date +%Y%m%d%H%M%S`
# Se o IP foi registrado em logon anterior
if [ -f /var/run/smbgate/$2 ] ; then
   USUARIO=`cat /var/run/smbgate/$2`
   echo "FORCADO:" $2 $USUARIO >>/var/log/smbgate/smbgate.log
   /usr/local/bin/netlogoff.sh $USUARIO
fi
# Se o usuario for autorizado (possuir script)
if [ -f /etc/smbgate/users/$1 ] ; then
# Se o usuario foi registrado em logon anterior
# Forca o logoff
if [ -f /var/run/smbgate/$1 ] ; then
   echo "FORCADO :" $1 >> /var/log/smbgate/smbgate.log
   /usr/local/bin/netlogoff.sh $1
fi
echo $2 > /var/run/smbgate/$1
echo $1 > /var/run/smbgate/$2
#log de inicio de liberacao de acesso aqui.
echo "LOGON  :" $2 $MOMENTO $1 >>/var/log/smbgate/smbgate.log
EXTIF='eth0'
COMMAND='-A'
ADDRESS=`cat /var/run/smbgate/$1`
/etc/smbgate/users/$1 $COMMAND $ADDRESS $EXTIF
fi
```

Figura A.5: *Script /usr/local/bin/netlogon.sh* (com base em Mattar (2005))

```
#!/bin/sh
# /usr/local/bin/netlogoff.sh
# uso:netlogoff.sh <username>
if [ -f /var/run/smbgate/$1 ]; then
   EXTIF='eth0'
   COMMAND='-D'
   ADDRESS=`cat /var/run/smbgate/$1`
   /etc/smbgate/users/$1 $COMMAND $ADDRESS $EXTIF 2> /dev/null
   MOMENTO=`date +%Y%m%d%H%M%S`
   echo "LOGOFF  :" $ADDRESS $MOMENTO $1 >> \
        /var/log/smbgate/smbgate.log
   rm -f /var/run/smbgate/$1 2> /dev/null
   rm -f /var/run/smbgate/$ADDRESS 2> /dev/null
fi
```

Figura A.6: *Script netlogoff.sh* (com base em Mattar (2005))

```
1  #!/bin/sh
2  # /etc/smbgate/users/nomedousuario
3  # uso:  nomedousuario <comando> <IP> <interface>
4  USER=`echo $0 | cut -c20-`
5  COMMAND=$1
6  ADDRESS=$2
7  EXTIF=$3
8  IP="10.1.1.9"
9  IPTABLES='/sbin/iptables'
10 $IPTABLES $COMMAND POSTROUTING -t nat -s $ADDRESS -o $EXTIF \
11       -m state --state NEW \
12       -j LOG --log-level alert \
13       --log-prefix $USER" " --log-ip-options
14 $IPTABLES $COMMAND POSTROUTING -t nat -s $ADDRESS -o $EXTIF \
15       -j SNAT --to $IP
16 exit 0
```

Figura A.7: *Script* para atualização das regras no *Iptables* (com base em Mattar (2005))

Apêndice B

Configuração do *SSH*

O texto a seguir apresentado neste apêndice é uma tradução livre de parte da documentação disponível em Mattar (2005).

"Pode-se necessitar executar o PDC em um computador e ter outro computador como o gateway*, por uma razão qualquer. Deve-se então configurar o* gateway *para aceitar* logins *autenticados através de RSA, a partir do PDC, sem senhas. Em* `http://www.openssh.org/` *existem informações sobre como configurar adequadamente o* SSH *no servidor e no cliente, com este objetivo.*

Recomendação acerca da utilização de autenticação criptografada

Deve-se ler a documentação do SSH *e certificar-se de haver entendido completamente o que se está fazendo quando se configura o* RSA *ou qualquer outro tipo de autenticação criptografada. Mas se a segurança não for uma prioridade, simplesmente pode-se seguir este exemplo e ir adiante.*

Geração das chaves

Para criar um par de chaves, os comandos abaixo devem ser executados na máquina PDC:

```
pdc:~# ssh-keygen -t rsa
```

Responde-se as questões e copia-se a chave pública criada para o gateway*. Normalmente a chave pública é copiada para* `~.ssh/id_rsa.pub`

```
pdc:~# cd .ssh
pdc:~# scp id_rsa.pub root@gateway:/root/.ssh/authorized_keys2
```

Script de *logon* usando *SSH*

Na Figura B.1 é apresentado um script `/etc/smbgate/users/usuario` *padrão, modificado para usar a autenticação criptografada* SSH

```
1  #!/bin/sh
2  #
3  COMMAND=$1
4  ADDRESS=$2
5  EXTIF=$3
6  IPTABLES='/sbin/iptables'
7  ssh root@gateway $IPTABLES $COMMAND POSTROUTING \
8      -t nat -s $ADDRESS -o $EXTIF -j MASQUERADE
```

Figura B.1: *Script* para atualização das regras no *Iptables* usando *SSH* (MATTAR, 2005)

Observa-se que o `iptables` *é executado através do* SSH *no* gateway. *Mais uma vez, deve-se consultar a documentação a respeito do servidor* SSH.*"*