UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

NÚCLEO DE COMPUTAÇÃO ELETRÔNICA

VICENTE ESPERANÇA PADRENOSSO

UTILIZANDO RECURSOS AVANÇADOS DE ACLs EM EQUIPAMENTOS DE REDES CISCO

RIO DE JANEIRO

2010

VICENTE ESPERANÇA PADRENOSSO

UTILIZANDO RECURSOS AVANÇADOS DE ACLs EM EQUIPAMENTOS DE REDES CISCO

Monografia apresentada para obtenção do título de Especialista em Gerência de Redes de Computadores no Curso de Pós-Graduação Lato Sensu em Gerência de Redes de Computadores e Tecnologia Internet do Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro – NCE/UFRJ.

Orientador:

Moacyr Henrique Cruz de Azevedo, M.Sc., UFRJ, Brasil

RIO DE JANEIRO

2010

VICENTE ESPERANÇA PADRENOSSO

UTILIZANDO RECURSOS AVANÇADOS DE ACLs EM EQUIPAMENTOS DE REDES CISCO

Monografia apresentada para obtenção do título de Especialista em Gerência de Redes de Computadores no Curso de Pós-Graduação Lato Sensu em Gerência de Redes de Computadores e Tecnologia Internet do Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro – NCE/UFRJ.

Aprovada em março de 2010.

Moacyr Henrique Cruz de Azevedo, M.Sc., UFRJ, Brasil

RESUMO

PADRENOSSO, Vicente Esperança. **UTILIZANDO RECURSOS AVANÇADOS DE ACLs EM EQUIPAMENTOS DE REDES CISCO.** Monografia (Especialização em Gerência de Redes e Tecnologia Internet). Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro. Rio de Janeiro, 2010.

Às vezes nos encontramos pensando em como gerenciar uma rede, mas raramente dispensamos algum tempo para analisar se o tipo de tráfego que passa através dessa rede está realmente de acordo com o que precisamos. Isto é, até que um problema de tráfego apareça. Planejar e manter uma rede envolve mais do que apenas certificar-se de que pacotes podem trafegar entre dispositivos na rede. Como um administrador de rede, você também precisa assegurar-se de manter a segurança e estabelecer políticas de negócio para permitir que somente o tráfego autorizado possa fluir em sua rede.

Sua organização também pode ter diretrizes que você precisa implementar, como por exemplo, aplicar Qualidade de Serviço *(Quality of Service - QoS)* em políticas de filas, identificar e filtrar o tráfego, ou diminuir o custo com a utilização de links de internet, sempre que possível. Resumindo, manter a conectividade é importante, mas, você também precisa manter a segurança, manter a robustez da rede, e seguir as políticas de negócio de sua empresa. Neste trabalho, falarei como alcançar esses objetivos usando *Cisco IOS Access Control Lists (ACLs)*. Apresentarei os tipos de ACLs, para que servem os tipos apresentados, como construir e aplicá-las de forma correta.

ABSTRACT

PADRENOSSO, Vicente Esperança. **UTILIZANDO RECURSOS AVANÇADOS DE ACLs EM EQUIPAMENTOS DE REDES CISCO.** Monografia (Especialização em Gerência de Redes e Tecnologia Internet). Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro. Rio de Janeiro, 2010.

Sometimes we find us thinking about as managing a net, but rarely do we spend some time to analyze if the type of traffic that passes through this net is really in accordance with what we need. That is, until traffic problem appears. To plan and to keep a net involve more than what only to certify itself of that packets can flow between devices in the net. As a net administrator, you also need to make sure to keep the security and to establish business politics to allow that only authorized traffic can flow in its net.

Your organization also can have lines of direction that you need to implement, as for example, to apply Quality of Service (QoS) in queue polices, to identify and to filter the traffic, or to reduce the cost with the use of Internet Links, whenever possible. Summarizing, to keep the connectivity is important, but, you also need to keep the security, to keep the robust of net, and to follow the business polices of your company. In this work, I will speak as to reach these objectives using Cisco IOS Access Control Lists (ACLs). I will present the types of ACLs, the use of the presented types, as to construct and to apply them of correct form.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACL	Access Control List
CBAC	Context-Based Access Control
CEF	Cisco Express Forwarding
DoS	Denial of Service
DSCP	Differentiated Services Control Point
FAB	Firewall ACL Bypass
FTP	File Transfer Protocol
ICMP	Internet Control Message Protocol
IOS	Internetwork Operating System
ISPs	Internet Service Providers (Provedores de Acesso à Internet)
NAT	Network Address Translation
QoS	Quality of Service (Qualidade de Serviço)
RACL	Reflexive Access List
SMTP	Simple Mail Transfer Protocol
TFTP	Trivial File Transfer Protocol
ToS	Type of Service (Tipo de Serviço)
VPN	Virtual Private Networks

SUMÁRIO

1 INTRODUÇÃO

Supondo vivermos no melhor de todos os mundos possíveis, os administradores de rede nunca precisariam aplicar políticas de rede. Crackers1 nunca atacariam um roteador para invadir uma rede, os roteadores nunca passariam informações de roteamento erradas, e os pacotes nunca tomariam os destinos que os administradores de rede não pretenderiam. Infelizmente, nós vivemos em um mundo hostil, imperfeito. Considere os seguintes cenários:

- Crackers invadindo a rede de Bancos e fazendo transações em sua conta. Ou invadindo a rede de sua empresa e roubando informações importantes para seus negócios.
- Uma empresa tem duas conexões de Internet com dois diferentes Provedores de Acesso a Internet (Internet Service Providers - ISPs)[2], ambos com mesma largura de banda. Isto foi implementado para prover uma rota alternativa caso uma das conexões falhar. Um dos ISPs é cobrado conforme o tráfego no link, ou seja, quanto maior a utilização maior será o custo, enquanto o outro tem preço fixo, independente do uso. Para reduzir custos, a companhia quer usar o ISP de preço fixo como link principal, a menos que esta conexão falhe, neste caso usará o link de internet baseado no tráfego. Devido não ter uma política de roteamento implementada para reforçar esta preferência, todo tráfego de Internet passa pela conexão de custo baseado no tráfego, forçando a companhia a incorrer em custos mais altos do que necessário.

Como exemplo, vamos analisar mais atentamente o primeiro cenário. Este cenário envolve Crackers que invadem um Web Site e alteram ou "roubam" informações. Permitir somente o acesso HTTP (acesso web), vindo da Internet, ao Servidor Web da empresa pode extremamente reduzir a probabilidade de uma invasão. Mesmo que alguém consiga acesso a este servidor, impedindo o uso de serviços tais como Telnet ou FTP, vindos da Internet ou de sentido contrário, ficaria difícil explorar o servidor como uma plataforma para novos ataques. Igualmente seria difícil transferir arquivos ou outras informações para o servidor. Um administrador de rede deve preocupar-se sobre problemas de segurança na rede: modificação não autorizada da informação, ataques de Negação de Serviço (Denial of Service – DoS)[3],

---

[1] ***Cracker*** é o termo usado para designar quem pratica a quebra (ou *cracking*) de um sistema de segurança, de forma ilegal ou sem ética. Este termo foi criado em 1985 por hackers em defesa contra o uso jornalístico do termo *hacker*.

[2] O **provedor de acesso à Internet** (em em inglês *Internet Service Provider*, **ISP**) oferece principalmente serviço de acesso à Internet, agregando a ele outros serviços relacionados, tais como *"e-mail"*, *"hospedagem de sites"* ou blogs, entre outros.

[3] Um **ataque de negação de serviço** (também conhecido como *Denial of Service* - **DoS**), é uma tentativa em tornar os recursos de um sistema indisponíveis para seus utilizadores. Alvos típicos são servidores web, e o

acesso não autorizado, e de interceptação de informações. Apresentarei como usar o Cisco IOS[4] Access Control List para reforçar políticas de segurança, e em outros casos também.

Para maioria dos exemplos, foi utilizado um laboratório composto por um roteador Cisco 1841, utilizando Cisco IOS Software, 1841 Software (C1841-IPBASE-M), Version 12.4(16b), RELEASE SOFTWARE (fc3), como mostrado na Figura 1, pelo comando ***show version***. Para outros exemplos foi utilizado o software Cisco Packet Tracer[5], versão 5.2.1.0006, Figura 2.

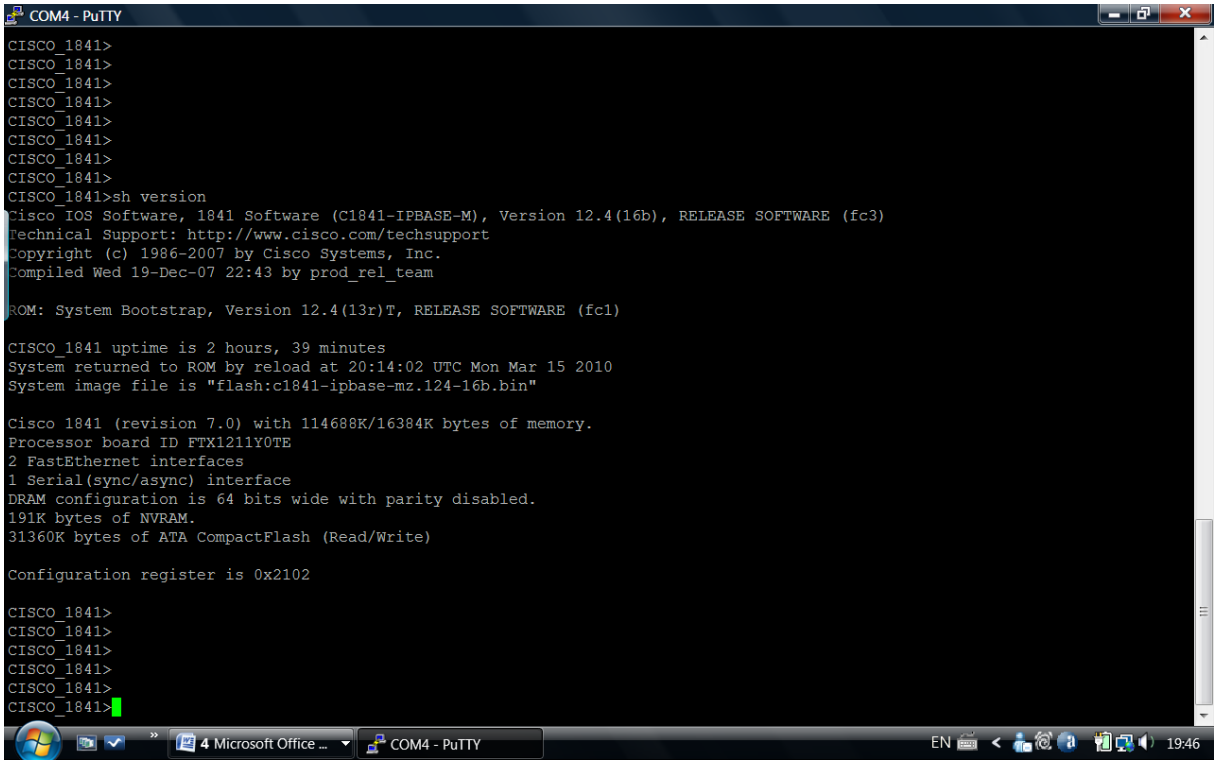

Figura 1 – Cisco 1841, comando ***show version***

---

ataque tenta tornar as páginas hospedadas indisponíveis na WWW. Não se trata de uma invasão do sistema, mas sim da sua invalidação por sobrecarga.

[4] O **Cisco IOS** (*Internetwork Operating System*) é o software usado na vasta maioria dos Roteadores e Switches Cisco (anteriormente era usado o CatOS). O IOS é um pacote com funções de roteamento, switching, integração entre redes e telecomunicações utilizando um sistema multitarefa. O primeiro IOS foi escrito por William Yeager.

[5] O **Packet Tracer** é um programa educacional gratuito desenvolvido pela empresa Cisco com o objetivo de simulação de rede de computadores, através equipamentos e configurações presente em situações reais. O programa apresenta uma interface gráfica simples, com suportes multimídia (gráfica e sonora) que auxiliam na confecção das simulações.

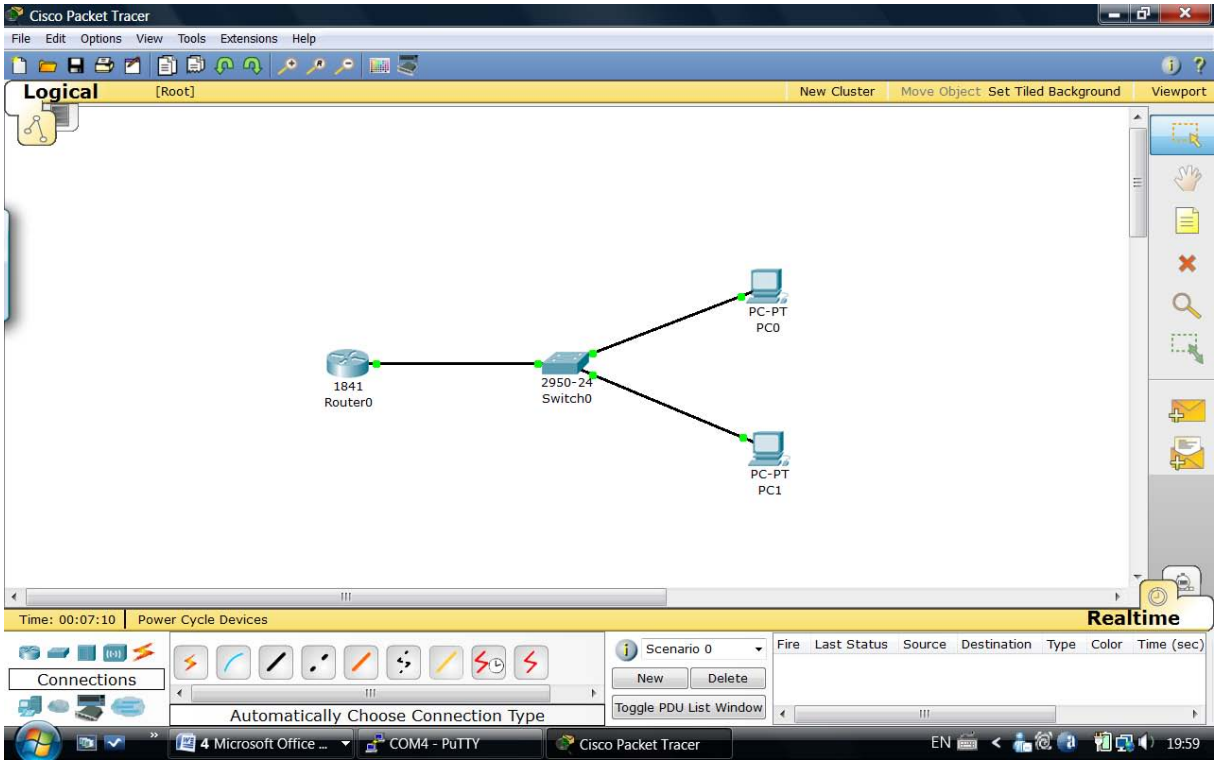

Figura 2 – Cisco Packet Tracer

1.1 USAR ACLs ou FIREWALLS?

Em questões de segurança, o que seria melhor, ACLs ou Firewalls[6]? Se os roteadores (ou switches que suportam ACLs) podem verificar milhões de pacotes por segundo entrantes, e podem verificá-los, ou filtrá-los, utilizando ACLs, até que ponto os Firewalls são bons? Por outro lado, a pergunta é: "Qual a diferença entre ACLs e Firewalls?" ou, "Onde posso aplicar ACLs?" A resposta depende do nível de proteção que você quer fornecer e o tipo de ataques que você poderá enfrentar. ACLs controlam quais protocolos e/ou portas um *host* pode usar para alcançar o objetivo. Por essa razão, ACLs são referidas frequentemente à camada 3 ou à camada 4.

Ao contrário da maioria dos Firewalls, ACLs comportam-se de maneira *stateless*, ou seja, todo o tráfego entrando o saindo é verificado pacote por pacote e comparado à ACL para que sejam tomadas as decisões de aceitar ou rejeitar o pacote, de acordo com as ações que o administrador determinou. Um Firewall *stateful*, por outro lado, verifica o tráfego entrante e compara a uma política (que é atualmente muito similar ao formato de uma ACL) e cria um

[6] Um **firewall** é parte de uma rede e é projetado para bloquear o acesso não autorizado e ao mesmo tempo permitir as comunicações autorizadas. É um dispositivo ou conjunto de dispositivos que está configurado para permitir ou negar o computador com a aplicação de um conjunto de regras e outros critérios. Podem ser implementados em hardware ou software, ou uma combinação de ambos.

registro da conexão se o tráfego for permitido. Os pacotes subsequentes que pertencem a esta conexão são permitidos automaticamente sem verificar novamente a regra criada, ou seja, a regra não é consultada de novo. Embora isto permita conseguir relatórios e *logs* detalhados (por exemplo, um Firewall permite de forma fácil fornecer o acesso e arquivos de log baseado por conexão), também pode nos trazer certas desvantagens. Imagine este cenário: um ataque que consiste em emitir um grande número de pacotes do Internet Control Message Protocol (ICMP). A última coisa que você quer neste caso é encher a tabela de conexões do Firewall do perímetro, ou borda. Neste cenário destaca-se uma característica específica dos Firewalls: Eles são *stateful*, mantêm o estado – o estado das conexões. Não é uma característica desejável manter o estado da conexão nestes casos, porque equipamentos *stateful* têm um limite de conexões simultâneas que podem lidar. Depois que a tabela de conexões estiver cheia, o tráfego que chega de forma legítima é negado devido ao dano causado pelo ataque. Este ataque é conhecido como Negação de Serviço (DoS). Este é um ponto onde os Firewalls perdem quando comparados a dispositivos *stateless*, tais como os roteadores e switches processando ACLs.

A escolha entre ACLs ou Firewalls nem sempre é necessária, na maioria dos casos um complementa o outro.

2 ACCESS CONTROL LISTS (ACLs)

2.1 INTRODUÇÃO

As Listas de Acesso não são somente um método para análise e filtro de pacotes. Porém, há muitas razões para se fazer este tipo de teste padrão, como por exemplo, por questões de segurança restringir o acesso a determinados recursos na rede. ACLs não são somente para filtrar pacotes. Podemos também usá-las para uma variedade de operações. O IOS dos roteadores suportam muitos tipos de *Access Control Lists (ACLs),* ou Listas de Controle de Acesso. As ACLs podem ser usadas para vários objetivos: existem ACLs que examinam critérios da Camada 2, tais como valores endereços MAC e LSAP. Existem ACLs que examinam vários protocolos de Camada 3, tais como IPX, AppleTalk, DECnet e Vines. Existem ACLs que examinam portas. Existem ACLs que examinam critérios IPv6. Existem ACLs para implementar QoS. Existem ACLs para controlar o acesso a links Internet, entre outros tipos de uso, indo dos mais simples aos mais complexos. Porém, ACLs também têm suas limitações, como por exemplo, não filtram o tráfego gerado e enviado para o próprio roteador. Portanto, o tráfego deve fluir de uma interface para outra para ser analisado. Seguem alguns exemplos de ACLs: *Standard ACL (ACLs Padrão), Extended ACLs (ACLs Estendidas), Numbered ACLs (ACLs* Numeradas), *Named ACLs (ACLs Nomeadas), Dynamic ACLs (lock-and-key), Reflexive ACLs, Time-based ACLs (time ranges), Commented IP ACL entries (remark), Context-based ACLs, Authentication proxy, Turbo ACLs, Distributed time-based ACL*. Estes são alguns dos tipos que podem ser configuradas no Cisco IOS. Dentro de cada uma destas categorias gerais, há muitos tipos diferentes de ACLs, que podem ser utilizadas de várias formas, das quais falarei com mais detalhes adiante.

Uma observação importante é que a Cisco® periodicamente adiciona novos grupos e tipos de ACLs ao IOS, portanto, versões mais antigas podem não suportar todos esses tipos citados. Também devemos compreender que se as características do IOS utilizado não suportar um protocolo em particular, como IPX ou AppleTalk, então o tipo correspondente de ACL também não estará disponível. Portanto, a maioria dos exemplos estarão baseados no IOS instalado no Roteador utilizado no laboratório (Cisco 1841), mas não deixarei de citar, de forma resumida, os outros tipos.

2.2 COMO USAR ACLs

Podemos aplicar uma ACL de muitas maneiras diferentes. Aplicada a uma interface, você pode usá-la para aceitar ou rejeitar os pacotes entrando ou saindo, baseadas em informações do protocolo, como por exemplo, endereço de origem e/ou destino, número da

porta, número do protocolo, e assim por diante. Aplicada a um protocolo de roteamento, uma ACL pode impedir que o roteador compartilhe informação sobre determinadas rotas em particular.

2.3 CONCEITOS BÁSICOS SOBRE ACLs

2.3.1 **ACLs Padrão x Estendidas**

Standard (padrão) e *Extended* (estendida) são os dois tipos principais de ACLs no Cisco IOS. A decisão quanto ao uso de ACLs padrão ou estendidas depende sobre qual é o seu objetivo. Por exemplo, se você simplesmente quer negar acesso a determinados computadores na rede, baseados nos seus endereços IPs, uma ACL padrão seria suficiente. Resumidamente, uma ACL IP padrão permite somente que o endereço de origem de um pacote seja usado nas decisões de filtragem.

Entretanto, se suas necessidades forem mais especificas, como por exemplo, você quer impedir que certos grupos de computadores acessem via Telnet um servidor em particular, uma ACL estendida seria necessária. ACLs estendidas permitem que o filtro seja aplicado mais detalhadamente. Por exemplo, uma ACL estendida permite que pacotes IP sejam filtrados de acordo com o endereço de origem, endereço de destino, tipo do protocolo, número de portas, e assim por diante.

2.3.2 **ACLs Numeradas x ACLs Nomeadas**

No Cisco IOS podem ser configurados diversos tipos diferentes de ACLs. Porém, as mais comuns são as ACLs numeradas, que podem ser padrão ou estendidas. As ACLs Padrão são numeradas no intervalo <1-99>, e as ACLs Estendidas numeradas no intervalo <100-199>. Estes intervalos foram expandidos para também incluir <1300-1999> para as ACL IP Padrão e <2000-2699> para as ACLs IP Estendidas. Estes números não somente identificam uma lista de acesso, mas também especificam o tipo de lista, baseado no *range* numérico a que faz parte. A lista mostrada na Figura 3 exibe os *ranges* numéricos associados aos diferentes tipos de ACLs. Dependendo dos protocolos suportados pelo seu IOS, a lista exibida pode ser diferente, mas o *range* numérico permanece o mesmo.

```
CISCO_1841(config)#access-list ?
<1-99>             IP standard access list
<100-199>          IP extended access list
<1100-1199>        Extended 48-bit MAC address access list
<1300-1999>        IP standard access list (expanded range)
<200-299>          Protocol type-code access list
<2000-2699>        IP extended access list (expanded range)
<700-799>          48-bit MAC address access list
compiled           Enable IP access-list compilation
dynamic-extended   Extend the dynamic ACL absolute timer
rate-limit         Simple rate-limit specific access list
```

Figura 3 – Exemplos de ACLs numeradas (Router Cisco 1841).

ACLs nomeadas podem ser usadas para comparar os mesmos pacotes, com os mesmos parâmetros, que você pode comparar com as ACLs IP numeradas - padrão e estendida. A mais óbvia diferença entre as ACLs numeradas e nomeadas é que o IOS identifica ACLs nomeadas usando nomes, ficando mais fácil gerenciá-las. ACLs nomeadas têm também outra característica muito importante, que as ACLs numeradas não têm: você pode deletar uma linha individual, sem necessidade de deletar toda a ACL. Em uma ACL numerada a exclusão de uma linha apaga toda a ACL criada. Uma ótima dica, que serve para os dois casos, é escrever toda ACL em um editor de textos, e aplicá-la depois de revisada em seu equipamento. Além disso, existe outra diferença importante de configuração entre ACLs numeradas e nomeadas. ACLs nomeadas usam uma configuração global que coloca o usuário em um submodo de configuração, onde são configuradas as permissões e negações lógicas (permit / deny). Na Figura 4 temos um exemplo de criação de uma ACL IP padrão nomeada.

```
CISCO_1841(config)#ip access-list ?
   extended    Extended Access List
   log-update  Control access list log updates
   logging     Control access list logging
   resequence  Resequence Access List
   standard    Standard Access List

CISCO_1841(config)#ip access-list standard ?
   <1-99>       Standard IP access-list number
   <1300-1999>  Standard IP access-list number (expanded range)
   WORD         Access-list name

CISCO_1841(config)#ip access-list standard ACL_NOMEADA
CISCO_1841(config-std-nacl)#?
  default  Set a command to its defaults
  deny     Specify packets to reject
  exit     Exit from access-list configuration mode
  no       Negate a command or set its defaults
  permit   Specify packets to forward
  remark   Access list entry comment
```

Figura 4 – Exemplo de uma ACL IP Padrão nomeada (Router Cisco 1841).

No exemplo mostrado na Figura 4 foi criada uma ACL nomeada, chamada ACL_NOMEADA. Observe que podemos usar o caractere "?" para exibir a listagem dos parâmetros que podem ser utilizados na construção da ACL. Depois de criada a ACL, repare que entramos no submodo de configuração da ACL criada – `CISCO_1841(config-std-nacl)#` – onde é possível configurar os parâmetros exibidos na lista. Falarei sobre cada um mais a frente. Como já citado anteriormente, dependendo da versão e dos protocolos suportados pelo seu IOS, a lista exibida pode ser diferente.

2.3.3 **Parâmetros Permit / Deny**

Existem apenas dois parâmetros quando configuramos uma ACL: *permit* ou *deny*.

Estes parâmetros são seguidos das definições: o que se deve permitir (*permit*) ou negar (*deny*). Estas são as mais básicas opções para as ACLs padrão e estendidas, numeradas ou nomeadas:

- ANY (tudo)
- HOST [IP do host]
- A.B.C.D [subrede + wildcard]
- [protocolo]

Ou seja, podemos especificar desde um único host até uma rede ou subrede inteira, podemos especificar um determinado protocolo, ou mesmo tudo. Na Figura 5 temos um exemplo dos parâmetros *permit* e *deny* na construção de uma ACL padrão. Na figura 6 são exibidos os parâmetros para construção de uma ACL Estendida.

```
CISCO_1841(config)#access-list 1 permit ?
  A.B.C.D  Address to match
  any      Any source host
  host     A single host address

CISCO_1841(config)#access-list 1 deny ?
  A.B.C.D  Address to match
  any      Any source host
  host     A single host address
```

Figura 5 – Parâmetros que podem ser configurados em uma ACL Padrão numerada.

```
Router(config)#access-list 101 permit ?
  eigrp  Cisco's EIGRP routing protocol
  gre    Cisco's GRE tunneling
  icmp   Internet Control Message Protocol
  ip     Any Internet Protocol
  ospf   OSPF routing protocol
  tcp    Transmission Control Protocol
  udp    User Datagram Protocol

Router(config)#access-list 101 deny ?
  eigrp  Cisco's EIGRP routing protocol
  gre    Cisco's GRE tunneling
  icmp   Internet Control Message Protocol
  ip     Any Internet Protocol
  ospf   OSPF routing protocol
  tcp    Transmission Control Protocol
  udp    User Datagram Protocol
```

Figura 6 – Parâmetros que podem ser configurados em uma ACL Estendida numerada.

2.3.4 **Máscara Coringa (*Wildcard Mask*)**

Uma coisa muito importante que deve ser lembrada quando queremos especificar um grupo de *hosts* em uma rede, ou uma rede inteira, são as máscaras coringas, ou *wildcard mask*, usadas pelas ACLs. O conceito de máscara coringa é muito similar aqueles de máscara de subrede, com uma pequena mas muito significante diferença na lógica. Uma máscara de subrede é uma sequência de binário 1s e 0s que são utilizados para determinar qual parte de um endereço IP é associado com o endereço de rede. Com uma máscara de subrede, se um bit em particular em um endereço IP é parte de um endereço de rede, o correspondente bit na máscara de subrede é setado para 1. Uma máscara coringa é muito similar, porém a lógica é inversa. Um binário 1 na máscara coringa significa “não se preocupe”. É o digito binário 0 na máscara coringa que distingue os bits importantes, que devem ser comparados. Em outras palavras pegamos a máscara de subrede (em decimal), convertemos para binário e invertemos os 0s e 1s, depois convertemos de novo para decimal.

Como exemplo, temos uma rede classe C – 192.168.10.0/24, portanto temos a máscara de rede 255.255.255.0 (ou em binário 11111111.11111111.11111111.00000000). Para chegarmos à máscara coringa temos que inverter os 0’s e 1’s, dessa forma temos em binário – 00000000.00000000.00000000.11111111. Quando convertido novamente em decimal temos: 0.0.0.255. Esta é a mascara coringa *(wildcard)* para rede 192.168.10.0/24. Da mesma forma podemos obter a máscara coringa para um determinado grupo *(range)* de host, e não somente para uma rede inteira. Por exemplo, para subrede 172.16.10.4/30. Serão testados os endereços 172.16.10.4 até 172.16.10.7.

	Endereço da subrede	Máscara de subrede	Máscara coringa
Decimal	172.16.10.4	255.255.255.252	0.0.0.3
Binário (último octeto)	00000100	11111100	00000011

Outro exemplo de forma bem resumida seria:

Suponha que você queira permitir o acesso de qualquer pacote que se origine na subrede 192.168.0.100 com máscara 255.255.255.240. Para encontrar a máscara coringa para esta máscara de subrede 255.255.255.240 basta fazer o seguinte:

- Onde na máscara se lê “255”, escreva “0”
- Onde na máscara se lê “0”, escreva “255”
- Onde for diferente de “0” e “255”, faça o cálculo “255-x”, onde “x” é o valor diferente.

Em nosso exemplo, o *wildcard* para a máscara de rede 255.255.255.240 é 0.0.0.15.

Também é necessário encontrar o endereço de subrede do IP 192.168.0.100 com máscara 255.255.255.240. As redes com essa máscara ocorrem de 16 em 16, portanto, a subrede deste IP seria 192.168.0.96. Assim sendo, uma ACL que permita o acesso de IPs originados na subrede em questão ficaria:

```
Router(config)# access-list 10 permit 192.168.0.96 0.0.0.15
```

2.3.5 **Deny Implícito**

Quando se trata de ACLs em equipamentos Cisco, trabalha-se com o conceito de negar tudo e permitir somente o que lhes for informado. Ou seja, toda ACL criada em um roteador Cisco termina com um DENY ANY implícito, negando TUDO, a todos. Por este motivo é preciso ter muito cuidado com a aplicação de uma ACL, pois se ela for mal planejada poderá isolar uma (ou várias) redes de uma só vez, inclusive impedindo o acesso ao roteador via Telnet, o que o impedirá de reverter a situação.

Observação importante: O que aconteceria se você apagasse uma ACL aplicada, por exemplo, a uma interface, sem antes removê-la desta interface?

A referência à lista de acesso permaneceria na interface. Porém, sem nenhuma entrada para ser comparada ao pacote entrante. Como sabemos que existe uma entrada *deny any* implícita no final de cada lista, seria esperado que uma ACL sem entradas negasse tudo. Mas, neste caso acontece o contrário. O comportamento de uma ACL vazia é permitir tudo.

Em uma ACL padrão, o comportamento seria:

```
access-list 1 permit any
```

Da mesma forma, o comportamento para uma ACL estendida, sem entradas, seria:

```
access-list 101 permit ip any any
```

2.4 CONSTRUINDO ACLs

2.4.1 **Sequência de Processamento das ACLs**

A análise feita pelo roteador sobre a ACL aplicada ocorre sempre de forma sequencial, de cima para baixo *(top-down).* Ou seja, as regras colocadas antes serão analisadas antes.

Uma vez que uma regra testada resulte em positivo nenhuma outra regra será analisada. Ou seja, se sua ACL tem 100 linhas, e supondo que o roteador ao comparar um pacote com as regras de sua ACL já der a primeira regra como positiva, nenhuma das outras 99 linhas serão examinadas e as ações definidas (permit ou deny) não serão tomadas. Portanto, lembre-se de sempre colocar as regras mais específicas antes e as mais genéricas depois. Do contrário as regras genéricas acabarão anulando as regras mais específicas. Por exemplo, se você quiser permitir toda a rede 192.168.20.0/24, mas negar um único endereço desta rede, como 192.168.20.10, você deverá colocar a entrada *deny* que é mais específica antes da entrada *permit* que é mais genérica, desta forma:

```
Router(config)#access-list 1 deny 192.168.20.10
Router(config)#access-list 1 permit 192.168.20.0 0.0.0.24
```

O comando ***show access-lists 1***, exibe a ACL criada e a ordem que as regras serão testadas:

```
Router#show access-lists 1
Standard IP access list 1
    deny host 192.168.20.10
    permit 192.168.20.0 0.0.0.24
```

Como já citado anteriormente (Item 2.3.5), devemos ter cuidado com a existência implícita de uma instrução *deny any* no final de cada ACL. O que significa que se não existirem coincidências das entradas anteriores para cada pacote, ou se você criar uma ACL contendo apenas *deny's*, sua ACL terá o mesmo efeito de dar um *shutdown* na interface onde ela for aplicada. Então, a classificação implícita será: negar todos os pacotes. Se desejar este *deny* implícito pode ser cancelado por uma entrada *permit any*.

2.4.2 **Aplicando ACLs**

Uma lista de acesso criada em um equipamento, por si só, não tem efeito algum. Para funcionar uma ACL precisa ser aplicada. Também temos que definir onde aplicá-la, que pode ser, por exemplo, em uma interface, linha vty (Telnet)[7] ou portas console e auxiliar, e a direção (entrada ou saída – *in ou out*) que devemos aplicá-la.

Como já mencionado, ACLs Padrão inspecionam apenas o endereço de origem no cabeçalho IP. Por este motivo, e seguindo as melhores práticas para aplicar ACLs, deve ser aplicada sempre o mais próximo do destino possível.

Para ACLs estendidas, seguindo as melhores práticas, devemos aplicá-la o mais próximo a origem. Isto porque elas podem especificar exatamente o tipo de tráfego, sua origem e seu destino, além de inspecionar o cabeçalho de segmentos encapsulados no pacote IP. Com isso evita-se que o tráfego não permitido tenha que atravessar a rede, ocupando processamento dos equipamentos e largura de banda, e ao chegar perto do seu destino "descobrir" que será descartado.

A regra básica diz que somente UMA ACL pode ser aplicada em uma mesma interface e direção, em um determinado roteador (ou switch), analisando pacotes de um determinado protocolo. Ou seja, em uma mesma interface você até pode ter mais de uma ACL desde que em sentidos opostos e/ou analisando protocolos diferentes. Os sentidos possíveis são ilustrados na Figura 7.

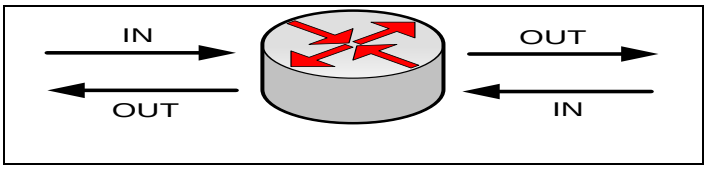

Figura 7 – Direções que as ACLs podem ser aplicadas.

O sentido em que uma ACL é aplicada determina qual o sentido do fluxo que deve ser examinado pelo roteador. Por este motivo, antes de aplicar uma ACL é necessário ter bem claro qual o efeito desejado.

Para melhor entendimento, segue abaixo um exemplo:

Na topologia apresentada na Figura 8, a seguinte ACL foi criada para negar ao host 204.204.7.89 acesso ao servidor FTP localizado em 196.6.13.254:

```
access-list 111 deny tcp host 204.204.7.89 host 196.6.13.254 eq 21
access-list 111 permit tcp any any
```

[7] (**V**irtual **T**elet**Y**pe) Uma interface de linha de comando criado em um roteador para uma sessão Telnet. O roteador é capaz de gerar um VTY dinamicamente.

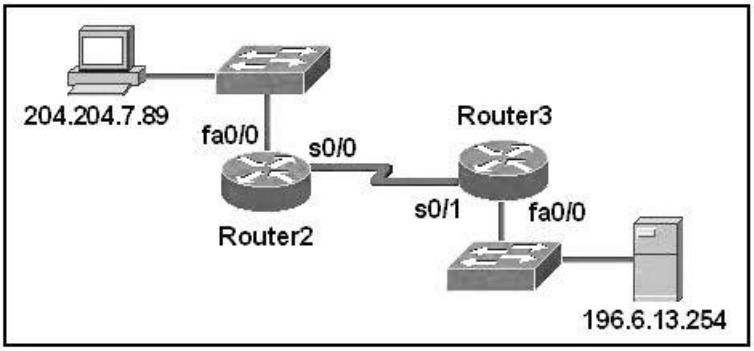

Figura 8 – Topologia 1.

Em qual interface e em qual direção devemos aplicá-la?

Como podemos observar, o Roteador mais próximo da origem do tráfego é o Router2 e, mais próxima ainda, sua interface FastEthernet 0/0. Como já mencionado, é aconselhável que as ACLs Estendidas sejam colocadas o mais próximo possível da origem do tráfego.

Aplicando a ACL na interface:

```
Router2(config)# interface fastEthernet 0/0
Router2(config-if)# ip access-group 111 in
```

Imagine, na mesma topologia apresentada, que se ao invés de usarmos uma ACL Estendida, usássemos uma ACL Padrão, ela ficaria assim:

```
access-list 1 deny host 204.204.7.89
access-list 1 permit any
```

Se aplicarmos essa ACL Padrão mais próxima a origem com sua direção correta, bloquearia o acesso deste host 204.204.7.89 não só ao servidor FTP, como também a qualquer outro destino. Logo, se fossemos utilizá-la, como recomendado pelas melhores práticas, seria melhor aplicá-la na saída da FastEthernet 0/0 do Router3, e, ainda assim, bloquearíamos o acesso deste host a outros possíveis hosts conectados a mesma interface F0/0 do Router3, o que, para o exemplo, não é o desejado. Então, apenas siga as melhores práticas para ACLs se elas fizerem sentido. Neste caso, não faz.

ACLs também podem ser aplicadas às linhas vty, a sintaxe é a seguinte:

```
Router(config)#line num_vty
Router(config-line)#access-class [numero-ACL] [in / out]
```

Aplicando uma ACL nomeada (TESTE) às linhas vty 0 4:

```
CISCO_1841(config)# line vty 0 4
CISCO_1841(config-line)#?
Line configuration commands:
  absolute-timeout      Set absolute timeout for line disconnection
  access-class          Filter connections based on an IP access list
  activation-character  Define the activation character
  !
  !
<--saída omitida-->

CISCO_1841(config-line)#access-class ?
  <1-199>      IP access list
  <1300-2699>  IP expanded access list
  WORD         Access-list name
CISCO_1841(config-line)#access-class TESTE in
CISCO_1841(config-line)#end
```

2.4.3 **Sintaxes Básicas Para Criação de ACLs**

Todas as ACLs são criadas em modo de configuração global, e depois de criadas devem ser aplicadas. Para os exemplos a seguir, usaremos a topologia mostrada na Figura 9.

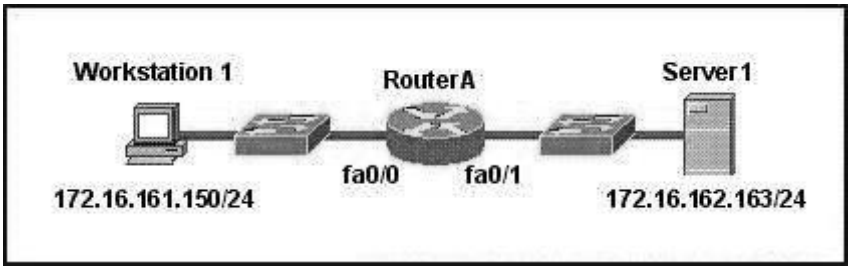

Figura 9 – Topologia 2.

2.4.3.1 ACLs Padrão Numeradas

```
access-list [num-access-list <1-99>] [deny/permit/remark]
[endereco-origem / host / any] [wildcard mask]
```

Exemplo:

Permitir somente que a Workstation 1 tenha acesso ao Server 1. Esta ACL pode ser criada de uma dessas três formas diferentes:

Criando a ACL

```
RouterA(config)#access-list 1 permit 172.16.161.150
ou
RouterA(config)#access-list 1 permit host 172.16.161.150
ou
RouterA(config)#access-list 1 permit 172.16.161.0 0.0.0.255
```

Devemos ter cuidado com efeito desejado ao criar esta ACL. As duas primeiras linhas têm o mesmo efeito: permitir somente o host 172.16.161.150, diferentes da última linha que permite toda rede 172.16.161.0/24.

Aplicando a ACL criada

```
interface [interface número]
ip access-group [número da lista] in / out

RouterA(config)#interface fastEthernet 0/1
RouterA(config-if)#ip access-group 1 out
```

2.4.3.2 ACLs Estendidas Numeradas

```
access-list [num-access-list <100-199>] [deny/permit/remark]
[protocolo-origem][endereco-origem / host / any] [wildcard mask-
origem] [endereco-destino / host / any] [wildcard mask-destino]
[operador[porta]]
```

Exemplo: Permitir que somente a Workstation1 tenha acesso permitido ao servidor WWW localizado no Server 1.

Criando a ACL:

```
RouterA(config)#access-list 100 permit tcp host 172.16.161.150
host 172.16.162.163 eq 80
```

Aplicando a ACL:

```
RouterA(config)#interface fastEthernet 0/0
RouterA(config-if)#ip access-group 1 in
```

2.4.3.3 ACLs Nomeadas (padrão e estendidas)

Como já falado no Item 2.3.2, neste tipo de ACL é permitido usar nome em vez de número de identificação, ficando mais fácil de ser gerenciada.

Para criar uma ACL nomeada utiliza-se a sintaxe:

```
Router(config)# ip access-list [standard / extended] NOME_ACL
```

Para aplicá-las são utilizados nomes, ao invés de números:

```
Router(config-if)#ip access-group NOME_ACL [in / out]
```

2.4.4 **Adicionando Comentários a uma ACL**

Podemos aplicar comentários às ACLs. Isto pode facilitar muito a vida dos administradores. ACLs podem ser longas e complexas. E, após algum tempo, pode ser difícil lembrarmos o porquê daquele conjunto de entradas esta lá. Ou pior ainda, imagine herdar de um administrador anterior uma ACL contendo centenas de entradas. Pode ser uma tarefa

muito dura ter que entender cada uma das linhas da ACL herdada. Por isso o uso dos comentários pode ser tão importante.

Estes comentários podem ser adicionados através do comando ***remark***, que foi introduzido na versão do Cisco IOS 12.0(2)T.

Podemos adicionar comentários em qualquer ACL numerada (padrão ou estendida), usando o comando ***remark:***

```
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#access-list 50 remark ACL para negar so o host 10.2.2.2
Router(config)#access-list 50 deny host 10.2.2.2
Router(config)#access-list 50 permit 10.2.2.0 0.0.0.255
Router(config)#access-list 50 permit any
Router(config)#end
```

Também podemos adicionar comentários em ACLs nomedas:

```
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#ip access-list standard ACL_TESTE
Router(config-std-nacl)#remark ACL para negar so o host 10.2.2.2
Router(config-std-nacl)#deny host 10.2.2.2
Router(config-std-nacl)#permit 10.2.2.0 0.0.0.255
Router(config-std-nacl)#permit any
Router(config-std-nacl)#end
```

O tamanho do comentário pode ter até 100 caracteres. Mas, se for necessário maior espaço pode-se simplesmente adicionar mais linhas *remarks* à ACL.

```
Router1(config)#access-list 50 remark ACL para negar o host 10.2.2.2
Router1(config)#access-list 50 remark Toda rede 10.2.2.0 sera permitida
Router1(config)#access-list 50 remark Todo o resto sera permitido
```

Os comentários não são exibidos pelo comando ***show access-lists***. A única forma de vê-los é exibindo a configuração do Roteador, com o comando ***show running-config***.

```
Router#show access-list 50
Standard IP access list 50
    deny   10.2.2.2
    permit 10.2.2.0 0.0.0.255
    permit any
Router#show running-config | include access-list 50
access-list 50 remark ACL para negar o host 10.2.2.2
access-list 50 remark Toda rede 10.2.2.0 sera permitida
access-list 50 remark Todo o resto sera permitido
access-list 50 deny   10.2.2.2
access-list 50 permit 10.2.2.0 0.0.0.255
access-list 50 permit any
Router#
```

3 USO DE PARÂMETROS AVANÇADOS

Neste capítulo falarei sobre as diversas formas e parâmetros disponíveis quando criamos ACLs. As sintaxes e os exemplos criados são baseados nos protocolos suportados pela versão do IOS instalado no Roteador utilizado no laboratório – Cisco 1841. Porém, não deixarei de comentar sobre outros tipos e parâmetros existentes. Anteriormente foram apresentados os parâmetros básicos das ACLs Padrão e Estendidas (*deny*, *permit, host, any, wildcard*). Agora veremos alguns tipos e parâmetros avançados.

3.1 FILTRANDO PELA APLICAÇÃO

ACLs IP estendidas também podem filtrar baseadas em aplicações, protocolos e número de portas.

```
Router1#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router1(config)#access-list 101 permit tcp any any eq www
Router1(config)#access-list 101 deny tcp any any gt 1023
Router1(config)#access-list 101 permit icmp any any
Router1(config)#access-list 101 permit udp any any eq ntp
Router1(config)#access-list 101 deny ip any any
Router1(config)#interface Serial0/1
Router1(config-if)#ip access-group 101 in
Router1(config-if)#exit
Router1(config)#end
Router1#
```

No exemplo mostrado na ACL numerada estendida (`access-list 101`) foram criadas entradas para permitir / negar o tráfego de protocolos como IP, TCP e UDP, baseadas em aplicações como WWW e também em números de portas (*eq* / *gt*). Após isso, foi aplicada a interface de entrada serial 0/1. Nos próximos tópicos segue a descrição mais detalhada de cada parâmetro.

3.1.1 **Aplicações TCP**

Podemos ver todos os parâmetros possíveis simplesmente usando a facilidade da ajuda online "?".

```
Router(config)#access-list 101 permit tcp ?
  A.B.C.D  Source address
  any      Any source host
  host     A single source host

Router(config)#access-list 101 permit tcp any ?
  A.B.C.D  Destination address
  any      Any destination host
  eq       Match only packets on a given port number
  gt       Match only packets with a greater port number
  host     A single destination host
  lt       Match only packets with a lower port number
  neq      Match only packets not on a given port number
  range    Match only packets in the range of port numbers


Router(config)#access-list 101 permit tcp any eq ?
  <0-65535>  Port number
  ftp        File Transfer Protocol (21)
  pop3       Post Office Protocol v3 (110)
  smtp       Simple Mail Transport Protocol (25)
  telnet     Telnet (23)
  www        World Wide Web (HTTP, 80)
```

3.1.2 **Aplicações UDP**

```
Router(config)#access-list 101 permit udp ?
  A.B.C.D  Source address
  any      Any source host
  host     A single source host


Router(config)#access-list 101 permit udp any ?
  A.B.C.D  Destination address
  any      Any destination host
  eq       Match only packets on a given port number
  gt       Match only packets with a greater port number
  host     A single destination host
  lt       Match only packets with a lower port number
  neq      Match only packets not on a given port number
  range    Match only packets in the range of port numbers


Router(config)#access-list 101 permit udp any eq ?
  <0-65535>  Port number
  bootpc     Bootstrap Protocol (BOOTP) client (68)
  bootps     Bootstrap Protocol (BOOTP) server (67)
  domain     Domain Name Service (DNS, 53)
  snmp       Simple Network Management Protocol (161)
  tftp       Trivial File Transfer Protocol (69)
```

3.2 RESTRINGINDO SESSÕES TCP (Parâmetro *Established*)

Um parâmetro adicional que pode ser utilizado em conexões TCP é o parâmetro *established*. Basicamente, permite que respostas vindas de conexões estabelecidas (*established)* sejam recebidas. O parâmetro não se preocupa em olhar se o tráfego de retorno em uma conexão estabelecida foi originado na rede interna. ACLs padrão e estendidas são filtros de pacotes simples, não mantêm informações sobre o estado de conexões. Portanto, o parâmetro *established* permite o recebimento do tráfego de retorno, mas compromete a segurança.

O parâmetro *established* checa se os *flags*[8] de controle ACK, FIN, PSH, RST, SYN, ou URG TCP estão setados, o Anexo 1 exibe uma breve explicação sobre cada *flag*. Se estiverem a ação para este tráfego TCP é executada (*permit* ou *deny*), se não, o Cisco IOS por padrão assume que é uma nova conexão originada na Internet e o tráfego é negado.

```
Router(config)#access-list 120 permit tcp any any ?
  dscp          Match packets with given dscp value
  eq            Match only packets on a given port number
  established   established
  gt            Match only packets with a greater port number
  lt            Match only packets with a lower port number
  neq           Match only packets not on a given port number
  precedence    Match packets with given precedence value
  range         Match only packets in the range of port numbers
  <cr>
```

No exemplo mostrado na Figura 10, o cliente terá permissão para iniciar uma sessão Telnet para o servidor:

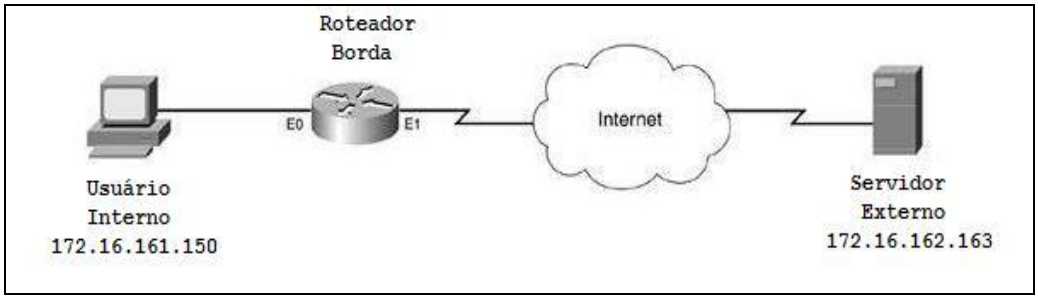

Figura 10 – Exemplo do parâmetro *established*

```
RouterA(config)#access-list 110 permit tcp host 172.16.161.150 eq
telnet host 172.16.162.163 established
RouterA(config)#access-list 110 deny ip any any
RouterA(config)#interface Ethernet0/0
RouterA(config-if)#ip access-group 110 in
RouterA(config-if)#exit
RouterA(config)#end
```

[8] As conexões TCP normais possuem um ou mais ***flags*** definidos- Os *flags* são usados para indicar a finalidade da conexão. O Anexo 1 mostra os *flags* TCP, sua representação e seu significado.

Neste exemplo, a interface aceitará pacotes TCP entrantes somente se o número da porta de origem for 23 (Telnet), e somente se esta sessão TCP for *established.* A palavra-chave *established* especifica somente permitir pacotes que estejam associados com uma conexão TCP estabelecida, ou resumidamente, após completado o TCP *three-way handshake*[9]. Isto não restringe o número da porta de destino porque o servidor inicialmente selecionou qualquer número aleatório de porta alta como sua porta de origem quando iniciada a sessão.

O que poderia acontecer sem uma conexão *established* presente? A interface aceitaria qualquer pacote TCP entrante com porta de origem 23. Mas, poderia ser literalmente qualquer coisa, e não realmente uma conexão Telnet. Qualquer *Cracker*, que saiba como alterar a porta de origem poderia facilmente iniciar uma conexão com o servidor no outro lado do roteador.

Então, se por questões de segurança você usa ACLs para controle de aplicações TCP, seria uma boa idéia considerar o uso do parâmetro *established.*

3.3 FILTRANDO APLICAÇÕES MULTIPORTAS

Alguns protocolos usam múltiplas portas. Os parâmetros *range, gt, lt e neq* podem ser usados para especificar um intervalo contíguo ou quais portas serão comparadas. Um exemplo clássico é o FTP, que usa as porta conhecidas 20 e 21, respectivamente, para controle (servidor) e envio de dados (cliente). Tomando este caso como exemplo, podemos diminuir as entradas de uma ACL, usando, por exemplo o parâmetro *range.* Desta forma, não seriam necessárias duas entradas para filtrar as portas 20 e 21 do FTP:

```
Router(config)#access-list 120 permit tcp any any eq 20
Router(config)#access-list 120 permit tcp any any eq 21
```

Neste exemplo, poderíamos usar:

```
Router(config)#access-list 120 permit tcp any any range 20 21
```

Similarmente, existem outros parâmetros para portas:

```
gt          Match only packets with a greater port number
lt          Match only packets with a lower port number
neq         Match only packets not on a given port number
```

- gt (greater than) – maior que
```
Router(config)#access-list 153 permit tcp any any gt 1023
```

- lt(less than) – menor que
```
Router(config)#access-list 153 permit tcp any any lt 1024
```

[9] O *three-way handshake* é o procedimento usado para estabelecer uma conexão quando o protocolo TCP é utilizado. O *three-way handshake* reduz as possibilidades de falsas conexões.

- neq (not equal to) – diferente de
```
Router(config)#access-list 153 permit tcp any any neq 666
```

Obviamente, antes da construção de toda ACL, temos que analisar o efeito desejado e de que forma aplicá-la.

3.4 CLASSIFICANDO O TRÁFEGO

Filtros podem ser aplicados em cabeçalhos de pacotes IP para implementação de QoS na rede. Podem ser baseados em *Differentiated Services Control Point* (DSCP), Precendência e *Type of Service* (ToS). (Para não fugir do objetivo, não entrarei em detalhes sobre a marcação dos pacotes, mas no Capítulo 4 - criação de ACLs avançadas, falarei sobre como criar ACLs utilizando esses parâmetros).

```
Router(config)#access-list 120 deny tcp any any ?
  ack           Match on the ACK bit
  dscp          Match packets with given dscp value
  eq            Match only packets on a given port number
  established   Match established connections
  fin           Match on the FIN bit
  fragments     Check non-initial fragments
  gt            Match only packets with a greater port number
  log           Log matches against this entry
  log-input     Log matches against this entry, including input interface
  lt            Match only packets with a lower port number
  neq           Match only packets not on a given port number
  precedence    Match packets with given precedence value
  psh           Match on the PSH bit
  range         Match only packets in the range of port numbers
  rst           Match on the RST bit
  syn           Match on the SYN bit
  time-range    Specify a time-range
  tos           Match packets with given TOS value
  urg           Match on the URG bit
  <cr>
```

3.4.1 **Filtros Baseados no Parâmetro DSCP**

Podemos fazer com que o roteador marque os campos *Differentiated Services Control Point* (DSCP) em pacotes IP para classificar sua prioridade através da rede.

Filtros podem ser aplicados baseados nos índices do campo do DSCP em um cabeçalho TCP, usando o parâmetro *dscp.* A tabela 1 exibe os valores DSCP.

```
Router(config)#access-list 120 deny tcp any any dscp ?
  <0-63>   Differentiated services codepoint value
  af11     Match packets with AF11 dscp (001010)
  af12     Match packets with AF12 dscp (001100)
  af13     Match packets with AF13 dscp (001110)
  af21     Match packets with AF21 dscp (010010)
  af22     Match packets with AF22 dscp (010100)
  af23     Match packets with AF23 dscp (010110)
  af31     Match packets with AF31 dscp (011010)
  af32     Match packets with AF32 dscp (011100)
  af33     Match packets with AF33 dscp (011110)
  af41     Match packets with AF41 dscp (100010)
  af42     Match packets with AF42 dscp (100100)
  af43     Match packets with AF43 dscp (100110)
  cs1      Match packets with CS1(precedence 1) dscp (001000)
  cs2      Match packets with CS2(precedence 2) dscp (010000)
  cs3      Match packets with CS3(precedence 3) dscp (011000)
  cs4      Match packets with CS4(precedence 4) dscp (100000)
  cs5      Match packets with CS5(precedence 5) dscp (101000)
  cs6      Match packets with CS6(precedence 6) dscp (110000)
  cs7      Match packets with CS7(precedence 7) dscp (111000)
  default  Match packets with default dscp (000000)
  ef       Match packets with EF dscp (101110)
```

Tabela 1 – DSCP – valores padrão

Drop Precedence	**Class 1**		**Class 2**		**Class 3**		**Class 4**	
	Value	Name	Value	Name	Value	Name	Value	Name
Lowest Drop Precedence	001010(10)	AF11	010010(18)	AF21	011010(26)	AF31	100010(34)	AF41
Medium Drop Precedence	001100(12)	AF12	010100(20)	AF22	011100(28)	AF32	100100(36)	AF42
Highest Drop Precedence	001110(14)	AF13	010110(22)	AF23	011110(30)	AF33	100110(38)	AF43

3.4.2 **Filtros Baseados no Parâmetro *precedence***

O parâmetro opcional precedência habilita o filtro baseado em um nível especifico, classificado de 0 a 7. Este campo no cabeçalho de um pacote IP tipicamente é usado para classificar pacotes para implementação de QoS e políticas de filas. Abaixo segue a lista dos nomes que podem ser usados ao invés dos números:

```
Router(config)#access-list 120 permit tcp any any precedence ?
  <0-7>          Enter up to 4 precedence values separated by white-spaces
  critical       Match packets with critical precedence (5)
  flash          Match packets with flash precedence (3)
  flash-override Match packets with flash override precedence (4)
  immediate      Match packets with immediate precedence (2)
  internet       Match packets with internetwork control precedence (6)
  network        Match packets with network control precedence (7)
  priority       Match packets with priority precedence (1)
  routine        Match packets with routine precedence (0)
```

Table 2 - Precedence - valores padrão

IP Precedence	Decimal value	Bit pattern
Routine	0	000
Priority	1	001
Immediate	2	010
Flash	3	011
Flash Override	4	100
Critical	5	101
Internetwork Control	6	110
Network Control	7	111

3.4.3 FILTROS BASEDOS NO PARÂMETRO ToS

O parâmetro, também opcional, *Type of Service* (ToS) habilita o filtro no campo ToS no cabeçalho de um pacote IP. Esta opção, como as outras apresentadas acima, também pode ser usada para implementar QoS. Os parâmetros são classificados de 0 a 15 ou pelos nomes dos serviços:

```
CISCO_1841(config)#access-list 101 permit tcp any any tos ?
  <0-15>              Type of service value
  max-reliability     Match packets with max reliable TOS (2)
  max-throughput      Match packets with max throughput TOS (4)
  min-delay           Match packets with min delay TOS (8)
  min-monetary-cost   Match packets with min monetary cost TOS (1)
  normal              Match packets with normal TOS (0)
```

Table 3 – TOS - valores padrão

IP TOS	Decimal value	Bit pattern
Normal	0	0000
Minimum monetary cost	1	0001
Maximum reliability	2	0010
Maximum throughput	4	0100
Minimum delay	8	1000

```
CISCO_1841(config)#access-list 101 permit tcp any any tos [ 0-15 |
max-reliability | max-throughput | min-delay | min-monetary-cost |
normal ]?
  ack           Match on the ACK bit
  eq            Match only packets on a given port number
  established   Match established connections
  fin           Match on the FIN bit
  fragments     Check non-initial fragments
  gt            Match only packets with a greater port number
  log           Log matches against this entry
  log-input     Log matches against this entry, including input interface
  lt            Match only packets with a lower port number
  neq           Match only packets not on a given port number
  precedence    Match packets with given precedence value
  psh           Match on the PSH bit
  range         Match only packets in the range of port numbers
  rst           Match on the RST bit
  syn           Match on the SYN bit
  time-range    Specify a time-range
  urg           Match on the URG bit
  <cr>
```

4 UTILIZAÇÃO AVANÇADA DE ACLs

4.1 CRIANDO ACLs PARA IMPLEMENTAR QoS

A razão principal para usar QoS em uma rede IP é proteger o tráfego sensível em links congestionados. Tudo que você pode fazer com um sistema de QoS sofre influência sobre quais pacotes serão encaminhados e quais serão descartados quando um congestionamento for encontrado. Isto é somente eficaz quando a congestão é intermitente. Se um link é utilizado de forma sobrecarregada, QoS oferecerá, no melhor dos casos, uma medida provisória até que o link seja atualizado ou a rede remodelada.

Essencialmente existem três etapas para qualquer esquema de priorização de tráfego. Primeiro estabelecer políticas para saber que tipo de tráfego é prioritário na rede, quais são críticos, quais podem esperar, etc. Após a identificação do tráfego prioritário, deve-se fornecer uma maneira de identificar, ou classificar como prioritário ou não, baseado na política adotada anteriormente para escolha. Utilizaremos os parâmetros citados anteriormente (Item 3.4) para classificar o tráfego e, finalmente, concluir a terceira e última etapa: configurar os dispositivos da rede (roteadores e switches) para usar essas informações e tomar as ações necessárias. Nesta etapa é decidido precisamente o que fazer com os tipos diferentes de tráfego e como serão encaminhados pela rede.

4.1.1 **Classificando Pacotes (campos DSCP ou ToS)**

Para os pacotes serem tratados deve existir alguma coisa que os priorizem de acordo com a forma desejada e conforme os diferentes tipos de tráfego. Antes de aplicar qualquer tratamento especial a um pacote, você deve ter claramente a idéia de qual tráfego precisa de tratamento especial e qual tratamento receberá. No exemplo a seguir será dado tratamento especial para o tráfego FTP recebido em uma interface serial. Este exemplo mostrará como "setar" um valor de Precedência IP *immediate* (2) para todo tráfego de controle FTP entrantes pela interface serial0/0, e um valor de Precedência IP *priority* (1) para todo tráfego de dados FTP. Isto é possível porque o tráfego de controle FTP usa a porta TCP 20, e tráfego de dados a porta 21. Para esta configuração também serão usados *class maps* e *policy-maps*. A Cisco introduziu esta característica na versão 12.0(5)T do IOS. Este método define uma *class-map*, que especifica como o roteador identificará este tipo de tráfego, e então definir um *policy-map*, que realmente aplique as mudanças no campo ToS dos pacotes.

```
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#access-list 101 permit any eq ftp any
Router(config)#access-list 101 permit any any eq ftp
Router(config)#access-list 102 permit any eq ftp-data any
Router(config)#access-list 102 permit any any eq ftp-data
Router(config)#

Router(config)#class-map ?
  WORD       class-map name
  match-all  Logical-AND all matching statements under this classmap
  match-any  Logical-OR all matching statements under this classmap
  type       type of the class-map
Router(config)#

Router(config)#class-map match-all ?
  WORD  class-map name
Router(config)#

Router(config)#class-map match-all ser00-ftpcontrole
Router(config-cmap)#?
  description  Class-Map description
  exit         Exit from class-map configuration mode
  match        classification criteria
  no           Negate or set default values of a command

Router(config-cmap)#
Router(config-cmap)#description trafego do controle ftp
Router(config-cmap)#match input-interface serial0/0
Router(config-cmap)#match access-group 101
Router(config-cmap)#exit
Router(config)#
Router(config)#class-map match-all ser00-ftpdados
Router(config-cmap)#description trafego de dados ftp
Router(config-cmap)#match input-interface serial0/0
Router(config-cmap)#match access-group 102
Router(config-cmap)#exit
Router(config)#

Router(config)#policy-map ?
  WORD  policy-map name

Router(config)#
Router(config)#policy-map serialftppolicy
Router(config)#
Router(config-pmap)#?
  description  Policy-Map description
  class        policy criteria
  exit         Exit from policy-map configuration mode
  no           Negate or set default values of a command

Router(config-pmap)#description ftp traffic policy
Router(config-pmap)#
```

```
Router(config-pmap)#class ?
  WORD           class-map name
  class-default  System default class matching otherwise unclassified packets
  type           type of the class-map

Router(config-pmap)#class ser00-ftpcontrole
Router(config-pmap-c)#?
  bandwidth       Bandwidth
  exit            Exit from class action configuration mode
  no              Negate or set default values of a command
  priority        Strict Scheduling Priority for this Class
  queue-limit     Queue Max Threshold for Tail Drop
  random-detect   Enable Random Early Detection as drop policy
  service-policy  Configure Flow Next
  set             Set QoS values
  shape           Traffic Shaping

Router(config-pmap-c)#set ip precedence immediate
Router(config-pmap-c)#exit

Router(config-pmap)#class ser00-ftpdados
Router(config-pmap-c)#set ip precedence priority
Router(config-pmap-c)#end
```

Aplicando a política do tráfego entrante da interface serial 0/0:

```
Router(config)#interface serial0/0
Router(config-if)#ip ?
<--saída omitida-->
!
  redirects     Enable sending ICMP Redirect messages
  rgmp          Enable/disable RGMP
  rip           Router Information Protocol
  route-cache   Enable fast-switching cache for outgoing packets
  rsvp          RSVP Interface Commands
  rtp           RTP parameters
  sap           Session Announcement Protocol interface commands
!
<--saída omitida-->

Router(config-if)#ip route-cache ?
  cef             Enable Cisco Express Forwarding
  flow            Enable Flow fast-switching cache
  policy          Enable fast-switching policy cache for outgoing packets
  same-interface  Enable fast-switching on the same interface
  <cr>

Router(config-if)#ip route-cache policy

Router(config-if)#service-policy ?
  input   Assign policy-map to the input of an interface
  output  Assign policy-map to the output of an interface

Router(config-if)#service-policy input ?
  WORD  policy-map name
```

```
Router(config-if)#service-policy input serialftppolicy
Router(config-if)#exit
Router(config)#end
Router#
```

4.1.2 **Usando Prioridade no Enfileiramento (parâmetro *precedence*)**

Podemos habilitar políticas de filas em interfaces para que o roteador possa sempre encaminhar primeiro pacotes com alta prioridade.

Para habilitar prioridade de fila em uma interface, primeiro deve-se estabelecer uma lista de prioridade e depois aplicar a uma interface.

Prioridade em filas assegura que pacotes com alta prioridade sejam encaminhados primeiro daqueles com baixa prioridade. Se não houver uma política claramente definida usar prioridade em filas pode ser uma má idéia porque os pacotes com alta prioridade podem tomar toda largura de banda disponível e os pacotes marcados com menor prioridade não receberem tratamento adequado. Segue abaixo um exemplo:

OBS: na primeira ACL (`access-list 101`) serão classificados com prioridade 5 (*precedence 5 - critical*) somente aos pacotes IP marcados com valor ToS igual a 12. Os pacotes não classificados terão baixa prioridade, como mostrado no comando ***priority-list 1 default low***.

```
Router(config)#access-list 101 permit ip any any precedence 5 tos 12
Router(config)#access-list 102 permit ip any any precedence 4
Router(config)#access-list 103 permit ip any any precedence 3
Router(config)#
Router(config)#priority-list ?
  <1-16>  Priority list number

Router(config)#priority-list 1 ?
  default      Set priority queue for unspecified datagrams
  interface    Establish priorities for packets from a named interface
  protocol     priority queueing by protocol
  queue-limit  Set queue limits for priority queues

Router(config)#priority-list 1 protocol ?
  arp            IP ARP
  bridge         Bridging
  cdp            Cisco Discovery Protocol
  compressedtcp  Compressed TCP (VJ)
  http           HTTP
  ip             IP
  llc2           llc2
  pad            PAD links
  pppoe          PPP over Ethernet
  snapshot       Snapshot routing support
```

```
Router(config)#priority-list 1 protocol ip ?
  high
  medium
  normal
  low

Router(config)#priority-list 1 protocol ip high list 101
Router(config)#priority-list 1 protocol ip medium list 102
Router(config)#priority-list 1 protocol ip normal list 103
Router(config)#priority-list 1 default low
Router(config)#interface Ethernet0
Router(config-if)#priority-group 1
Router(config-if)#exit
Router(config)#end
Router#
```

Parâmetros precedence:

```
Router(config)#access-list 101 permit tcp any any precedence ?
  <0-7>           Enter up to 4 precedence values separated by white-spaces
  critical        Match packets with critical precedence (5)
  flash           Match packets with flash precedence (3)
  flash-override  Match packets with flash override precedence (4)
  immediate       Match packets with immediate precedence (2)
  internet        Match packets with internetwork control precedence (6)
  network         Match packets with network control precedence (7)
  priority        Match packets with priority precedence (1)
  routine         Match packets with routine precedence (0)
```

Parâmetros ToS:

```
Router(config)# access-list 101 permit tcp any any precedence 5 tos ?
  <0-15>             Type of service value
  max-reliability    Match packets with max reliable TOS (2)
  max-throughput     Match packets with max throughput TOS (4)
  min-delay          Match packets with min delay TOS (8)
  min-monetary-cost  Match packets with min monetary cost TOS (1)
  normal             Match packets with normal TOS (0)
```

4.2 PROTEGENDO O ACESSO SNMP

Podemos prover segurança extra ao acesso SNMP usando ACLs.

Por padrão, quando habilitamos o serviço SNMP o roteador permite que todos os endereços IP acessem o agente SNMP na porta padrão UDP 161. É recomendável o uso de ACLs para permitir acesso SNMP a somente poucos hosts ou subredes.

```
Router(config)#ip access-list standard ACL_SNMP
Router(config-std-nacl)#permit 172.20.1.0 0.0.0.255
Router(config-std-nacl)#permit host 10.1.1.1
Router(config-std-nacl)#deny any
Router(config-std-nacl)#snmp-server community TESTE ro ACL_SNMP
Router(config)#end
Router#
```

O comando ***show snmp group*** exibe as ACLs associadas às comunidades SNMP.

```
Router>show snmp group
groupname:  TESTE security model:v1
readview :  v1default writeview: <no writeview specified>
notifyview: <no notifyview specified>
row status: active      access-list: ACL_SNMP
```

4.3 CONFIGURANDO NAT

Network Address Translation (NAT), chamado às vezes de *Network Address Translator (NAT),* foi descrito pela primeira vez na RFC 1631 em 1994. Os autores do documento estavam tentando resolver o problema do então iminente fim dos endereços IPv4. A proposta foi uma solução simples, mas brilhante. A idéia era permitir que os dispositivos no interior de uma rede usassem o conjunto (*pool*) padrão de endereços IP não registrados, que são definidos atualmente na RFC 1918. Então o roteador ou o firewall entre a rede privada interna e a rede pública externa teriam o software que reescrevesse (ou traduzisse) os IP internos em cada pacote substituindo-os pelos endereços válidos registrados.

ACLs são usadas para "selecionar" quais endereços internos serão traduzidos, de forma dinâmica, em endereços válidos. No exemplo da Figura 11, foi criada uma ACL padrão para traduzir todos os endereços da rede interna 192.168.0.0/16, para o *range* de endereços válidos 172.16.1.100 até 172.16.1.150

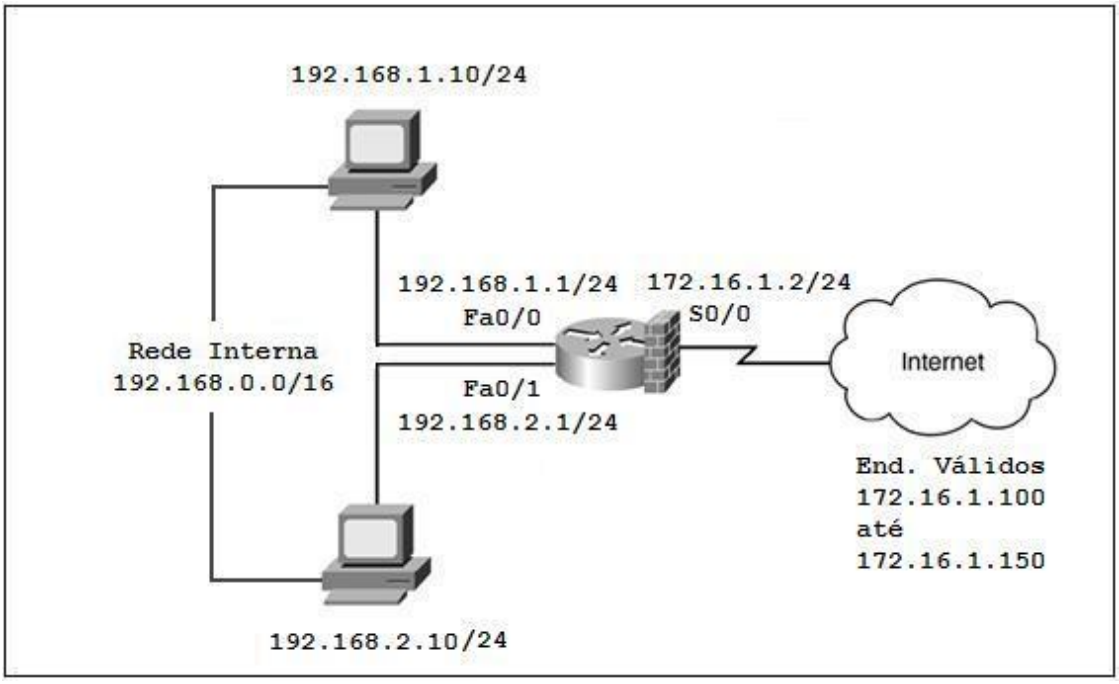

Figura 11 – Exemplo NAT

```
Router(config)#access-list 15 permit 192.168.0.0 0.0.255.255
Router(config)#ip nat pool NATPOOL 172.16.1.100 172.16.1.150 netmask
255.255.255.0
Router(config)#ip nat inside source list 15 pool NATPOOL
Router(config)#interface FastEthernet 0/0
Router(config-if)#ip address 192.168.1.1 255.255.255.0
Router(config-if)#ip nat inside
Router(config-if)#exit
Router(config)#interface FastEthernet 0/1
Router(config-if)#ip address 192.168.2.1 255.255.255.0
Router(config-if)#ip nat inside
Router(config-if)#exit
Router(config)#interface serial0/0
Router(config-if)#ip address 172.16.1.2 255.255.255.0
Router(config-if)#ip nat outside
Router(config-if)#exit
Router(config)#end
Router#
```

5 TIPOS AVANÇADOS DE ACLs

5.1 TURBO ACLs

Turbo ACLs permitem que os dispositivos possam melhorar o uso das ACLs, aumentando o desempenho no acesso e execução durante o processo de uma ACL.

Normalmente ACLs fazem uma busca seqüencial (de cima para baixo) para achar uma regra a ser comparada. ACLs são ordenas levando em consideração este fator. Durante essa tarefa ACLs adicionam uma quantidade substancial de tempo e memória quando pacotes estão sendo comparados e encaminhados. Além disso, o tempo gasto pelo roteador durante a procura de uma regra em uma lista adiciona atraso no encaminhamento de pacotes. Uma grande carga no processamento é necessária durante a procura em uma ACL com várias entradas.

Quando uma ACL é compilada ela se transforma em tabelas *lookup*[10] que podem ser processadas muito mais eficientemente do que pelo método linear padrão. Porém esta característica está somente disponível em roteadores de alta capacidade (como 7100, 7200, 7500, e a série 12000). Ao habilitar o uso de Turbo ACL, as ACLs configuradas são escaneadas e, se for o caso, compiladas. Este escâner e compilação podem demorar alguns segundos se o sistema estiver processando uma ACL complexa, ou quando o sistema estiver processando uma configuração que contém uma grande quantidade de ACLs.

Cabeçalhos de pacotes são usados para acessar essas tabelas menores, com número fixo, independente do número de entradas de uma ACL.

Compilar ACLs também exige uma memória RAM maior do que as não-compiladas. Com Turbo ACL, tabelas construídas na memória do roteador ajudam a acelerar o processamento do tráfego. Para ativar esta característica em um roteador compatível, emita o comando ***access-list compiled*** no modo de configuração global. Isto compilará todas as ACLs e diminuirá o atraso dos pacotes, porque o tempo gasto na comparação de pacotes é fixo. Qualquer modificação para uma ACL resultará em imediata recompilação daquela ACL. Para visualizar estatísticas das ACLs compiladas, como por exemplo, quantidade de memória utilizada, emita o comando ***show access-lists compiled***.

OBS: todas as ACLs criadas depois de ser emitido o comando ***access-list compiled***, também serão compiladas.

Ativando Turbo ACL:

```
CISCO_1841(config)#access-list compiled
```

[10] Uma **tabela *lookup*** é uma estrutura de dados, geralmente usada para organizar entradas e acelerar a execução com uma simples operação de indexação.

O comando ***show access-lists*** mostra as ACLs compiladas:

```
CISCO_1841#show access-lists
Standard IP access list 1 (Compiled)
    10 permit 192.168.1.1
    20 deny   192.168.2.0, wildcard bits 0.0.0.255
    30 deny   192.168.3.0, wildcard bits 0.0.0.255
    40 permit any
Standard IP access list 2 (Compiled)
    30 permit 172.16.3.1
    10 permit 172.16.1.0, wildcard bits 0.0.0.255
    20 permit 172.16.2.0, wildcard bits 0.0.0.255
    40 deny   172.16.3.0, wildcard bits 0.0.0.255
    50 permit any
Standard IP access list 23 (Compiled)
    10 permit 10.10.10.0, wildcard bits 0.0.0.7
Extended IP access list 101 (Compiled)
    10 permit tcp any any eq www
    20 permit icmp 172.20.10.0 0.0.0.255 host 172.20.5.1
    30 deny tcp any any eq telnet
    40 deny tcp any any eq ftp
```

Comando ***show access-lists compiled***, exibe estatísticas das ACLs compiladas:

```
CISCO_1841# show access-lists compiled
Compiled ACL statistics:
 ACL         State        Entries  Config  Fragment  Redundant
1           Operational      4        4         0         0
23          Operational      1        1         0         0
2           Operational      5        5         0         0
101         Operational      5        4         1         0
4 ACLs, 4 active, 10 builds, 15 entries, 156 ms last compile
0 history updates, 2000 history entries
0 mem limits, 128 Mb limit, 1 Mb max memory
0 compile failures, 0 priming failures
Overflows: L1 0, L2 0, L3 0
Table expands:[9]=0 [10]=0 [11]=0 [12]=0 [13]=0 [14]=0 [15]=0
L0: 1803Kb    4/5      6/7      9/10     3/4      3/4      2/3
5/6      4/5
L1:    5Kb    2/35     2/40     2/12     2/30
L2:    2Kb    2/150    2/150
L3:    2Kb    2/300
Ex:    7Kb
Tl: 1821Kb  50 equivs (14 dynamic)

Memory chunk statistics: (number passed/number failed)
  30/0 chunk creates, 27/n/a chunk destroys
  0/0* interrupt level, 60/0 process level allocations
  * failures at interrupt level do not indicate a memory shortage
  0/0 replenishes, 750/0 elements replenished *
  * including element allocation at chunk creation time
  0 online, 0 offline replenish suspends
```

Estes são os parâmetros para o comando ***show access-lists compiled***:

```
CISCO_1841#sh access-lists compiled ?
  |  Output modifiers
  <cr>

CISCO_1841#sh access-lists compiled | ?
  append    Append redirected output to URL (URLs supporting append operation
(only)
  begin     Begin with the line that matches
  exclude   Exclude lines that match
  include   Include lines that match
  redirect  Redirect output to URL
  section   Filter a section of output
  tee       Copy output to URL

CISCO_1841#sh access-lists compiled | tee ?
  /append  Copy and append output to URL (URLs supporting append operation
           only)
  flash:   Uniform Resource Locator
  ftp:     Uniform Resource Locator
  http:    Uniform Resource Locator
  nvram:   Uniform Resource Locator
  rcp:     Uniform Resource Locator
  tftp:    Uniform Resource Locator

CISCO_1841#sh access-lists compiled | section ?
  LINE     Regular Expression
  exclude  Exclude entire section(s) of output
  include  Include entire section(s) of output

CISCO_1841#sh access-lists compiled | redirect ?
  flash:  Uniform Resource Locator
  ftp:    Uniform Resource Locator
  http:   Uniform Resource Locator
  nvram:  Uniform Resource Locator
  rcp:    Uniform Resource Locator
  tftp:   Uniform Resource Locator

CISCO_1841#sh access-lists compiled | [include / exclude / begin] ?
  LINE  Regular Expression
```

OBS: se ainda não houver ACLs compiladas, esta será a saída para o comando ***show access-lists*** compiled:

```
CISCO_1841#show access-lists compiled
Compiled ACLs unavailable
```

5.2 REFLEXIVE ACLS

Reflexives ACLs (RACLs), ou ACLs Reflexivas, foram introduzidas pela primeira vez no Cisco IOS 11.3. Diferente das ACLs padrão, que pode filtrar informações de camada 3, e ACLs estendidas, que filtram informações das camadas 3 e 4, RACLs podem filtrar

informações das camadas 3, 4 e 5 (camada de sessão). RACLs têm várias vantagens, bem como algumas desvantagens. Geralmente são usadas quando não se tem acesso a *Context-based Access Control* (CABC), que é um *statefull firewall* que possui algumas funcionalidades a mais do que as RACLs (apresentarei mais à frente).

As ACLs padrão e estendidas proporcionam um eficiente filtro contra diversos tipos de ataques, mas, encontramos também certas limitações, principalmente quanto à segurança. A maior limitação dessas ACLs é que são um excelente filtro em conexões unidirecionais, mas não bidirecionais. ACLs padrão e estendidas não mantêm controle sobre informações do estado de uma conexão, ou seja, não sabem sobre o andamento de uma conexão (a não ser uma conexão TCP usando o parâmetro *established*). Em outras palavras, ACLs padrão ou estendidas filtram baseadas em informações estáticas que foram previamente configuradas.

RACLs mantêm informações sobre o estado de conexões originadas internamente e permitem, de forma dinâmica, que respostas para essas conexões retornem. Por padrão RACLs negam todo tráfego originado fora da rede em direção à rede interna. RACLs criam dinamicamente entradas provisórias no filtro de entrada sempre que algum dispositivo interno abre uma sessão para fora da rede. Esta entrada temporária permite o retorno do tráfego para o dispositivo interno através do roteador de borda, e é excluída logo que a conexão for encerrada, no caso de conexões TCP, ou que exceda o tempo limite, caso dos protocolos sem conexão como UDP.

5.2.1 **Diferenças Entre ACLs Estendidas e RACLs.**

Para ajudar a entender as vantagens de uma RACL sobre ACLs estendidas, neste tópico farei uma comparação.

Como citado anteriormente, as ACLs estendidas filtram baseadas nos endereços de origem e destino, informações da camada 3 e 4, como os protocolos IP (IP, ICMP, TCP, UDP, e outros), e outras informações baseadas em campos dos cabeçalhos nas camadas 3 e 4. O exemplo da Figura 12 mostra como uma ACL estendida lida com o tráfego ICMP originado quando um usuário interno (172.20.10.1) “pinga” (ICMP *echo-request*) um servidor externo (200.200.1.1), e a resposta (ICMP *echo-reply*) retorna.

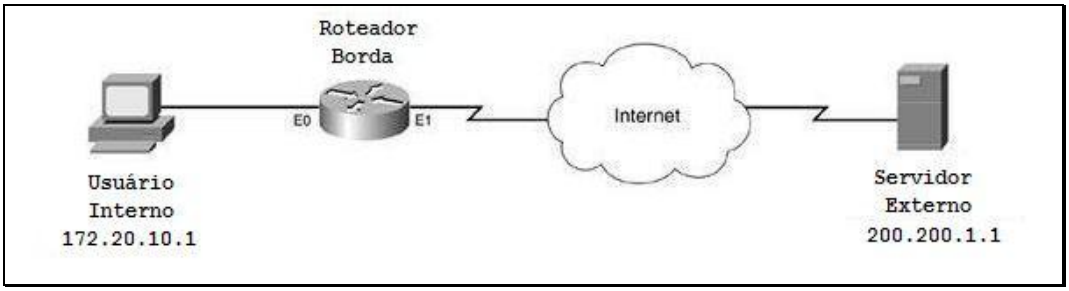

Figura 12 – Exemplo do funcionamento de uma ACL estendida

Assuma que o roteador de borda na Figura 12 permita a saída desse tipo de tráfego e, após alguns *pings* serem enviados para fora da rede, as respostas retornam ao roteador de borda. Para permitir que as respostas ICMP retornem ao destino na rede interna deve ser configurada uma ACL como esta:

```
Router(config)# ip access-list extended permit-icmp
Router(config-ext-nacl)# permit icmp any host 172.20.10.1 echo-reply
Router(config-ext-nacl)# deny ip any any
Router(config-ext-nacl)# exit
Router(config-ext-nacl)# interface ethernet1
Router(config-ext-nacl)# ip access-group permit-icmp in
```

O exemplo, mostrado na Figura 12, permite respostas ICMP para 172.20.10.1. Mas, com essa solução aparecem alguns problemas:

- Como não sabemos quem 172.20.10.1 está pingando, devemos permitir todas as respostas *echo-reply* para este host, dessa forma a rede fica aberta, e outros servidores além de 200.200.1.1 podem responder.
- Esta regra estará sempre ativa, assim um *Cracker* pode implementar uma ataque DoS.
- Este filtro serve somente para um único host: 172.20.10.1. Se for necessário permitir outros hosts, devemos incluí-los na regras da ACL.

5.2.2 **Como Funcionam as RACLs**

O exemplo anterior mostrou como ACLs estendidas lidam com as respostas para os pedidos gerados dentro da rede. Portanto, a solução apresentada abre um grande “buraco” na segurança da rede.

Ao contrário das ACLs estendidas, RACLs têm o controle sobre o estado de uma conexão, pelo menos em certo grau. Em outras palavras, uma RACL pode detectar quando um usuário inicia uma conexão para fora da rede e pode permitir a resposta à conexão deste usuário. ACLs estendidas podem fazer isso, mas somente para conexões TCP (com o

parâmetro *established*). RACLs podem fazer isso para todos os protocolos IP, porém, ao contrário das ACLs estendidas, somente permitem o tráfego de retorno quando uma sessão é iniciada dentro da rede. RACLs usam regras temporárias inseridas no filtro de uma ACL estendida, que é aplicada à interface externa do roteador de borda. Quando a sessão termina (no caso de uma conexão TCP) ou expira o tempo dessa entrada na ACL, ela é removida da ACL aplicada na interface externa do roteador. Isto reduz o risco de um ataque DoS. RACLs têm suas limitaçoes, mas pelo menos, possibilitam uma solução muito melhor do que só usar ACLs estendidas para prover segurança em roteadores de borda.

5.2.3 **Etapas de Processamento do Tráfego**

RACLs necessitam do uso de pelo menos duas ACLs para funcionar corretamente. A primeira ACL é usada para capturar informações para o tráfego de saída. Esta informação é colocada em uma RACL especial e inserida na ACL que foi aplicada na interface externa do roteador de borda. A Figura 13 mostra as etapas de processamento de uma RACL:

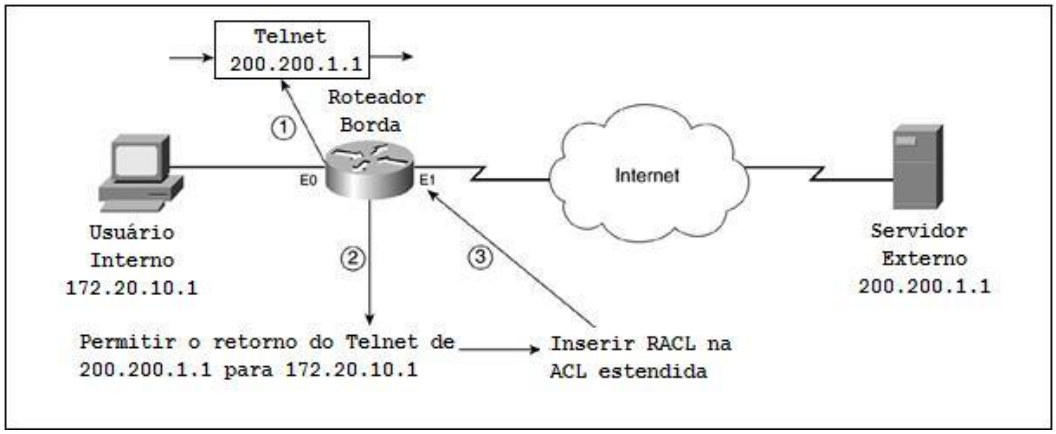

Figura 13 – Etapas de processamento de uma RACL.

Estas foram as etapas mostradas na Figura 13:

1. A ACL interna analisa o tráfego para iniciar uma sessão externa. Neste caso, a ACL estendida, que foi configurada previamente, permite que a rede interna tenha acesso Telnet para rede externa.
2. Entradas são adicionadas a RACL nomeada temporária (é uma ACL separada).
3. A RACL criada é inserida em uma terceira ACL aplicada na entrada da interface externa do roteador de borda (E1 – in). Esta RACL permitirá o retorno do tráfego para esta conexão.

Uma entrada é automaticamente removida quando a sessão e terminada, no caso de sessões TCP, ou quando expira o tempo desta entrada na RACL (fica ociosa por um período de tempo), no caso de sessões sem conexão. Por padrão, o tempo que uma entrada inserida em uma RACL permanece até que seja excluída são 300 segundos (5 minutos). É recomendado que o valor padrão seja alterado para sessões sem conexão, como UDP e ICMP. Em outras palavras, não seria seguro que uma requisição DNS ou um *ping* permaneça em uma RACL por 5 minutos, quando este tipo de sessão, geralmente, é finalizada em menos de 5 ou 10 segundos. Portanto, deixando esta entrada ativa por 5 minutos aumentaria o risco de sofrer algum tipo de ataque.

5.2.4 **Construindo RACLs (parâmetros *reflect* e *evaluate*)**

Quando um tráfego originado na rede interna sai em direção à Internet, certamente existirá uma ACL nomeada estendida para examiná-lo. Esta ACL pode ser aplicada no sentido entrante na interface interna ou na saída da interface externa.

Nesta ACL, que podemos chamá-la de ACL "interna", são criadas regras que examinam o tráfego para novas sessões (uso do parâmetro ***reflect***). Com estas regras, é possível controlar para quais conexões serão criadas entradas na RACL que permitirão que o tráfego retorne para a rede interna. Se estas regras não forem criadas, por padrão, o tráfego será negado. Isto proporciona maior controle sobre quais usuários internos terão acesso à rede externa e quais os tipos de conexões serão permitidas. Por exemplo, poderia ser criada uma ACL para permitir somente conexões HTTP, permitindo apenas serem criadas entradas em RACLs temporárias para o tráfego HTTP.

Para usar RACLs para filtrar o tráfego de retorno, deve ser criada uma ACL nomeada estendida para filtrar o tráfego vindo da rede externa. Esta ACL é aplicada no sentido entrante da interface externa ou na saída da interface interna e normalmente é chamada ACL "externa".

Na ACL externa deve ser colocada uma referência onde a RACL ou RACLs devem ser processadas. Basicamente, é um "ajustamento" da RACL com uma ACL. Qualquer regra criada na RACL será inserida logicamente na ACL externa no local especificado pela referência. Quando o Cisco IOS processa a ACL externa e encontra uma referência RACL, ele processa as regras na RACL até encontrar uma regra que se ajuste à comparação. Se não existir nenhuma regra que se ajuste na RACL, o IOS volta para a próxima regra depois da referência RACL e continua o processamento.

Um ponto importante é que uma RACL é ligeiramente diferente de uma ACL. Em uma RACL não existe a regra *deny any* implícita ao final da lista. Isto ocorre porque uma regra RACL é inserida em uma ACL externa, que deve ter outras regras abaixo dessa regra inserida.

A inserção de uma referência RACL é feita utilizando o parâmetro ***evaluate***. O posicionamento deste parâmetro dentro de uma ACL externa é muito importante. Por exemplo, segue uma configuração para uma ACL externa (podemos fazer referência a Figura 12, utilizada anteriormente, porém, supondo o tráfego www):

```
Router(config)# ip access-list extended ENTRADA
Router(config-ext-nacl)# permit tcp any host 172.20.10.1 eq www
Router(config-ext-nacl)# deny ip any any
Router(config-ext-nacl)# exit
Router(config)# interface fastethernet 0/1
Router(config)# description Interface de conexao com a Internet
Router(config)# ip access-group ENTRADA in
```

Neste exemplo somente é permitido tráfego web para o servidor interno (172.20.10.1), todo outro tipo de tráfego é negado.

Para permitir o retorno do tráfego na rede interna para sessões listadas na RACL temporária, deve ser colocada uma referência na ACL externa de entrada que direcione para o uso das regras da RACL. Deve-se lembrar que o local para inserir esta referência é importante. A referência deve ser colocada antes de qualquer regra *deny*, usando o parâmetro *evaluate*. Desta forma, o posicionamento correto seria:

```
Router(config)# ip access-list extended ENTRADA
Router(config-ext-nacl)# remark inserir a referência RACL aqui
Router(config-ext-nacl)# permit tcp any host 172.20.10.1 eq www
Router(config-ext-nacl)# deny ip any any
Router(config-ext-nacl)# exit
Router(config)# interface fastethernet 0/1
Router(config)# description Interface de conexao com a Internet
Router(config)# ip access-group ENTRADA in
```

Segue abaixo um exemplo simples, que mostra a utilização dos parâmetros *reflect* e *evaluate*:

Neste exemplo é negado todo tráfego originado na Internet. Porém, deve ser permitido todo tráfego TCP e UDP retornando da Internet para usuários internos.

```
Router(config)# ip access-list extended ACL_interna
Router(config-ext-nacl)# permit tcp any any reflect RACL_tcp
Router(config-ext-nacl)# permit udp any any reflect RACL_udp timeout 60
Router(config-ext-nacl)# exit
Router(config)# ip access-list extended ACL_externa
Router(config-ext-nacl)# evaluate RACL_tcp
Router(config-ext-nacl)# evaluate RACL_udp
Router(config-ext-nacl)# deny ip any any
Router(config-ext-nacl)# exit
```

```
Router(config)# interface fastethernet 0/1
Router(config)# description Interface de conexao com a Internet
Router(config-if)# ip access-group ACL_interna out
Router(config-if)# ip access-group ACL_externa in
Router(config-if)# exit
Router(config)#
```

5.2.5 **Conclusão**

Reflexive ACLs (RACLs) permitem o tráfego de retorno em direção a rede interna, quando iniciada uma sessão internamente. RACLs têm vantagens sobre o parâmetro *established* (Item 3.2) e sobre ACLs estendidas porque as entradas de uma RACLs são temporárias e desaparecem depois que o tempo de uma conexão ociosa expirar ou uma conexão TCP finalizar ou ser abortada, aumentando a segurança. O objetivo de uma *Reflexive ACL* é melhorar a uso das ACLs estendidas. RACLs avaliam o estado de uma conexão e não devem ser utilizadas em substituição aos Firewalls.

5.3 CONTEXT-BASED ACCESS CONTROL (CBAC)

Este tópico enfoca *Context-Based Access Control* (CBAC), uma das principais características do Cico IOS Firewall[11]. Foi introduzido no Cisco IOS 12.0(5)T. A Cisco recomenda o uso de CBAC ao invés de ACLs Reflexivas. CBAC fornecem muito mais funcionalidades e têm menos limitações que as RACLs. Somente algumas plataformas incluem suporte para o Cisco IOS Firewall, como roteadores série: SOHO70, SOHO90, 800, uBR900, 1600, 1700, 2500, 2600, 3200, 3600, 3700, 7100, 7200, 7300, 7400, 7500, e 7600. Switches incluem o Catalyst séries 4000, 5000, 6000, e 8850.

CBAC fornece filtragem na camada de aplicação, incluindo suporte para protocolos mais complexos e aplicações multimídia. Pode examinar conexões para NAT e PAT e fazer as traduções necessárias. Além disso, pode abrir conexão *stateful* adicional para suporte a aplicações, como FTP e H.323[12]. Seguem abaixo alguns protocolos a aplicações suportadas:

- Todas as conexões TCP e UDP, incluindo FTP, HTTP com Java, SMTP, TFTP, e comandos UNIX como *rexec*, *rlogin* e *rsh*;

---

[11] CBAC é apenas uma das muitas características do Cisco IOS Firewall. O **Cisco IOS Firewall** também inclui *authentication proxy* e *Intrusion Detection System* (IDS), entre outros, que não são assuntos para este trabalho.

[12] O padrão H.323 é parte da família de recomendações ITU-T (International Telecommunication Union Telecommunication Standardization sector) H.32x, que pertence a série H da ITU-T, e que trata de "Sistemas Audiovisuais e Multimídia".

- Conexões ICMP, incluindo *echo request, echo reply, destination unreachable, time exceeded, timestamp request*, e mensagens *timestamp reply*;
- Sun Remote Procedure Calls (RPCs);
- Oracle SQL *Net;
- Real-Time Streaming Protocol (RSTP), incluindo aplicações com RealNetworks, Cisco IP/TV e Apple Quick Time 4;
- H.323 v1 e v2, incluindo White Pine CU-SeeMe, Netmeeting e Proshare;
- Outras aplicações multimídia, incluindo StreamWorks, NetShow e VDOLive;
- Protocolos Voz sobre IP (VoIP), incluindo Session Initiation Protocol (SIP) e Skinny Client Control Protocol (SCCP);

5.3.1 **Funcionalidades CBAC**

CBCA disponibiliza quatro funções principais: filtro de tráfego, inspeção de tráfego, detecção de intrusos e geração de alertas e auditoria.

5.3.1.1 Filtro de Tráfego

Uma das principais funções do CBAC é filtrar inteligentemente, especialmente para conexões TCP, UDP e ICMP. Como RACLs, uma das funções é permitir que respostas retornem para a rede interna, entretanto, pode ser usado para filtrar tráfego originado em ambas as direções: interna e externa, ou seja, entrando ou saindo da rede interna.

Diferente das ACLs estendidas, que filtram somente nas camadas 3 e 4, e RACLs, que inclui a camada 5 (sessão), CBAC suporta inspeção de aplicação, isto é, pode examinar o conteúdo de certos tipos de pacotes. Por exemplo, pode examinar comandos SMTP em uma conexão SMTP.

5.3.1.2 Inspeção de Tráfego

Como mencionado na sessão anterior, CBAC pode inspecionar informações da camada de aplicação e usar para manter sua função de “firewall dinâmico”, mesmo para aplicações que abrem múltiplas conexões ou para endereços “traduzidos” (NAT).

Este processo de inspeção não somente permite a reposta que retorna para rede interna, mas também pode ser usado para prevenir ataques, como por exemplo, ataques de inundação, conhecidos como TCP SYN *flood*: CBAC pode examinar a taxa em que estas conexões estão sendo feitas para um serviço e pode cancelar estas conexões se um determinado limite é

atingido. Pode também examinar conexões TCP para certificar qual número de sequência cai em um determinado intervalo, negando qualquer pacote suspeito.

5.3.1.3 Detecção de Intrusos

Além de inspecionar o tráfego para implementar um firewall dinâmico, como visto anteriormente, CBAC também pode usar essa característica para detectar certos tipos de ataques DoS. CBAC pode fornecer proteção contra ataques ao servidor SMTP, limitando os tipos de comandos SMTP que podem ser enviados ao servidor interno. Todos esses tipos de ataques podem ser usados pelo CBAC para gerar informações de *log* (registro) sobre o ataque, bem como, opcionalmente, redefinir conexões TCP ou descartar pacotes maliciosos.

5.3.1.4 Geração de Alertas e Auditoria

CBAC pode gerar em tempo real alertas dos problemas e detecção de ataques, bem como fornecer auditoria detalhada dos pedidos de conexão. Por exemplo, todos os pedidos de conexão de rede podem ser registrados, incluindo os endereços IP de origem e destino, os números das portas usadas na conexão, o número de bytes enviados, e a hora em que a conexão começou e terminou.

5.3.2 **Funcionamento do CBAC**

Para manter o controle sobre conexões que está monitorando, o CBAC cria uma tabela de estados que contém informações sobre cada conexão. CBAC monitora conexões TCP, UDP e ICMP e mantém informações na tabela de estados para essas conexões. Então, CBAC usa a tabela para criar entradas ACLs dinamicamente para permitir que respostas retornem à rede interna através do roteador de borda. Isto é um pouco similar à RACLs, entretanto, CBAC pode inspecionar as informações da camada de aplicação, enquanto RACLs não podem. CBAC usa a tabela de estados e as entradas, dinâmicas, criadas em ACLs para prevenir certos tipos de ataques DoS, especialmente aqueles que envolvem TCP *flooding*.

No exemplo da Figura 14, seguem as etapas de um processamento CBAC:

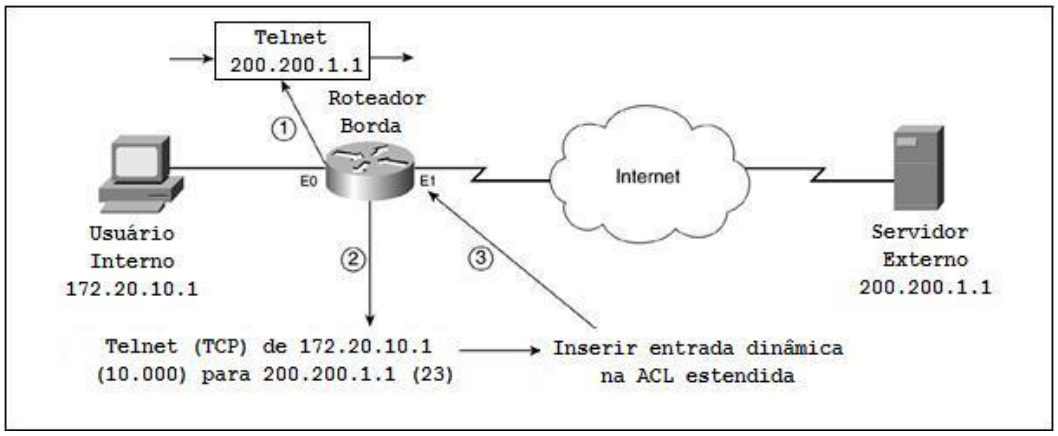

Figura 14 – Etapas do processamento CBAC

1. Um usuário interno inicia uma conexão de saída, como mostrado no exemplo, Telnet. Se uma ACL de entrada é aplicada (E0), esta é processada antes de qualquer inspeção por CBAC. O Cisco IOS, baseado em suas regras de inspeção para CBAC, pode ou não inspecionar a conexão. Se não inspecionar a conexão o IOS permite o pacote passar, caso contrário, o IOS processa a etapa 2.
2. O Cisco IOS compara esta conexão com a tabela de entradas: se esta conexão não existir a entrada é adicionada. Se existir o IOS "reseta" o temporizador para esta conexão.
3. Se esta for uma nova entrada o Cisco IOS adiciona uma entrada dinâmica em uma ACL estendida na entrada da interface externa (E1), para permitir o tráfego de retorno para rede interna. Estas entradas dinâmicas na ACL aplicada na interface externa não são salvas na NVRAM.

Este processo é muito similar a como RACLs são processadas. Como as RACLs, CBAC abre entradas temporárias em ACLs predefinidas para permitir o tráfego de retorno. Estas entradas são criadas quando o tráfego inspecionado internamente deixa a rede interna, e são removidas assim que a conexão é terminada ou o período de tempo ocioso é atingido. Tal como RACLs, pode ser especificado qual protocolo ou quais protocolos serão inspecionados, bem como, em qual interface e direção a inspeção deve ocorrer.

5.3.3 **Melhorias do CBAC em Relação às RACLs**

Um novo recurso foi introduzido no Cisco IOS 12.3(4)T, chamado *Firewall ACL Bypass* (FAB). Este recurso foi desenvolvido para acelerar o processo de retorno do tráfego para a rede. Com o recurso FAB, o IOS não cria entradas dinâmicas nas ACLs para permitir o tráfego de retorno. Em vez disso, o IOS examina a tabela de estados para determinar qual tráfego deve ser permitido retornar à rede, que pode ser tratado por um processo *fast*

switching, como Cisco Express Forwarding (CEF)[13]. Se o Cisco IOS não achar uma correspondência na tabela de estados o IOS usa a ACL aplicada na entrada da interface de retorno para impor políticas. Usando este processo o roteador não tem que criar e gerenciar entradas dinâmicas criadas na ACL aplica na interface que recebe o tráfego de retorno. Em vez disso o roteador tem somente que verificar a tabela de estados que ele mantém. Então, entradas dinâmicas em ACLs não são mais necessárias para permitir o tráfego de retorno. O recurso FAB é usado automaticamente e não pode ser desativado.

Outras melhorias também foram aplicadas:

- Tráfego TCP – O tráfego TCP é processado da mesma maneira como em RACLs. Uma única coisa que CBAC tem a mais do que RACLs é que CBAC também monitora o *setup* de uma conexão. Com CBAC, o Cisco IOS espera que a conexão seja estabelecida dentro de 30 segundos (este tempo é configurável pelo usuário). Se a conexao não for estabelecida durante este período, o IOS remove a entrada da sua tabela de estados e da ACL.
- Tráfego UDP – É assumido que se o tráfego UDP estiver ocioso por mais de 30 segundos (este tempo é configurável pelo usuário) a conexão deve ter concluída, então o IOS remove a entrada da tabela de estado e da ACL. O processo é muito similar com RACLs, exceto pelo tempo predefinido de *timeout*. CBAC tem uma melhoria no tráfego UDP sobre RACLs: também pode inspecionar requisições e respostas DNS. Com esse recurso, CBAC espera que quando um dispositivo interno gera uma consulta DNS, o servidor DNS remoto irá responder dentro de 5 segundos (este tempo é configurável pelo usuário). Se a resposta não chegar em 5 segundos a entrada da conexão DNS é removida, para prevenir um ataque. Da mesma forma, quando a resposta vinda do servidor DNS é recebida, o IOS imediatamente remove a entrada da sua tabela de estados e da ACL. Estas melhorias são usadas para prevenir ataques DNS *spoofing*[14] e DoS.
- Tráfego ICMP – Inspeção do tráfego ICMP foi introduzida no Cisco IOS 12.2(11)YU e foi integrada no 12.2(15)T. Antes disso, CBAC só podia inspecionar o tráfego TCP e UDP. CBAC pode inspecionar tipos de mensagens ICMP comuns, incluindo *echo request, echo reply, destination unreachable, time exceeded, timestamp request,* e

[13] **CEF** é utilizada principalmente para acelerar a comutação de pacotes, reduzindo a sobrecarga e atrasos introduzidos por outras técnicas de roteamento, aumentando o desempenho geral. CEF é composto por dois componentes principais: o Forwarding Information Base (FIB) e adjacências.

[14] **DNS *spoofing*** é um termo usado quando um servidor DNS aceita e usa informações incorretas vindas de um host não autorizado a dar essa informação.

timestamp reply. CBAC não inspeciona outros tipos de mensagens ICMP. O Cisco IOS espera respostas aos tipos de mensagens ICMP suportadas dentro de 10 segundos.

- Inspeção da Aplicação – CBAC também inspeciona informações na camada de aplicação para limitar interação entre dois dispositivos. Um bom exemplo disso é o apoio para inspeção SMTP. Para limitar a exposição de um sistema de emails, CBAC pode inspecionar comandos SMTP enviados entre dois servidores de email sobre a conexão de controle. Isto é usado para prevenir ataques de acesso. Com este recurso, CBAC permite somente certos comandos SMTP. Se qualquer outro comando SMTP é enviado ele será negado. Neste caso o Cisco IOS responde ao remetente com uma mensagem SMTP NOOP.
- Prevenção de Ataques DoS – CBAC pode detector certos tipos de ataques DoS. Quando um ataque ocorre, o IOS pode tomar qualquer das seguintes ações: bloquear os pacotes, proteger os recursos internos para que não fiquem sobrecarregados com conexões falsas ou gerar mensagens de alerta. Para detectar ataques DoS, CBAC usa limites de tempo (*timeout*) e valores de entradas para inspecionar o estabelecimento de conexões TCP.

5.3.4 **Limitações do CBAC**

Mesmo com todos os recursos e melhorias, CBAC não é a última solução para implementação de um firewall. Em outras palavras, CBAC tem suas limitações e não pode proteger a rede de todos os tipos de ataques. Entender as limitações do CBAC e do Cisco IOS irá ajudar a entender melhor se esta será a melhor solução para a rede ou se será um complemento para solução de segurança já existente.

Algumas limitações do CBAC são:

- Ele inspeciona apenas o tráfego que o administrador especificar. Isto é uma vantagem, mas também uma desvantagem. Ele permite controlar a sobrecarga que o CBAC coloca no roteador, bem como o tráfego que é permitido retornar à rede. Para torná-lo um produto abrangente, no entanto, o administrador precisa configurar muitos parâmetros de inspeção para cobrir todos os tipos de conexões.
- CBAC não é simples de entender e implementar. Ele requer conhecimento detalhado de protocolos e aplicações, bem como o seu funcionamento.
- Como acontece com ACLs, o Cisco IOS não pode usar o CBAC para inspecionar o tráfego originado pelo roteador.

- CBAC não pode inspecionar pacotes encriptados, como IPSec[15]. No entanto, se uma conexão VPN termina no roteador, ele pode inspecionar o tráfego entrando e saindo do túnel VPN criptografado.
- CBAC não suporta inspeção de todos os aplicativos. Para certos aplicativos funcionarem, a inspeção deve ser desabilitada.
- CBAC ignora mensagens ICMP *destination unreachable* (destino inacessível).
- O Cisco IOS não suporta redundância para a tabela de estados. Por exemplo, se um roteador falhar pode existir outro roteador como redundância, mas a tabela de estados não é replicada entre os dois. Neste exemplo, a tabela de estados deve ser reconstruída no segundo roteador, causando falhas em algumas conexões e exigindo que os usuários restabeleçam estas conexões.

5.3.5 **Configurando CBAC**

Ao contrário das configurações de RACLs, mostradas no item 5.2, a configuração de CBAC é mais complexa. Existem muito mais comandos com mais opções que podemos configurar. Para simplificar, o processo de configuração foi dividido em sete etapas principais, que seguem abaixo:

1. Determinar quais interfaces serão externas ou internas no roteador.
2. Criar uma ACL normal para filtrar o tráfego entrando ou saindo, certificando de que o tráfego que será inspecionado está saindo da rede.
3. Alterar os valores de *timeout* para as conexões (opcional).
4. Configurar *Port Application Mapping* (PAM) no qual especifica os números de portas que o CBAC deve inspecionar se a aplicação não estiver usando o número de porta padrão, como por exemplo, HTTP com porta 8080. Esta configuração é opcional e só é necessária se o aplicativo não estiver usando o número da porta padrão.
5. Definir as regras de inspeção. Estas regras definem quais entradas serão adicionadas à tabela de estados e qual tráfego de retorno será permitido. Se o tráfego de saída não corresponde a uma regra de inspeção o roteador não irá inspecioná-lo e irá tratar como tráfego normal.

[15] Protocolo de Segurança IP (IP Security Protocol, mais conhecido pela sua sigla, **IPSec**) é uma extensão do protocolo IP que visa a ser o método padrão para o fornecimento de privacidade do usuário, integridade dos dados e autenticidade das informações, quando se transferem informações através de redes IP pela internet.

6. Ativar a regra ou regras nas interfaces do roteador. O roteador então usará CBAC para inspecionar o tráfego.
7. Testar a configuração enviando pacotes passando pelo roteador CBAC. Mesmo que opcional, é recomendável. É a única forma de encontrar problemas.

Abaixo descreve-se as etapas citadas:

Etapa 1 – Seleção de Interfaces.

A primeira etapa é determinar qual interface é interna e qual é externa. Neste contexto, interna é onde é originada uma conexão e externa é onde receberá o tráfego de retorno dessas conexões. Se houver necessidade de configurar um filtro CBAC nas duas direções, é recomendado que primeiro configure somente em uma direção e aplique os testes antes de configurar a segunda direção. Configurar filtros em duas direções é comum em ambientes de intranets e extranets.

Etapa 2 – Configuração da ACL.

Nesta etapa são configuradas as ACLs para filtrar o tráfego entrando e possivelmente saindo da rede. A recomendação é que primeiro seja criada uma ACL básica para permitir somente o necessário dentro e fora da rede. Depois adicionam-se outros filtros para uma solução de segurança mais robusta. Isto simplifica o processo de resolução de problemas.

Depois de criar as ACLs, é necessário aplicá-las às interfaces do roteador. Entretanto, existe uma restrição sobre o uso de ACLs e sua aplicação quando estiver usando CBAC. Na entrada da interface externa e saída da interface interna, somente ACLs estendidas podem ser aplicadas. Em qualquer outra situação pode-se usar ACLs padrão ou estendidas. Isto é, na entrada da interface interna e na saída da interface externa, podem ser aplicadas tanto ACLs padrão como estendidas nas duas direções: *in* ou *out*. Ao contrário das RACLs, ACLs CBAC podem ser numeradas ou nomeadas (RACLs só podem ser nomeadas). A Figura 15 mostra o uso apropriado de ACLs.

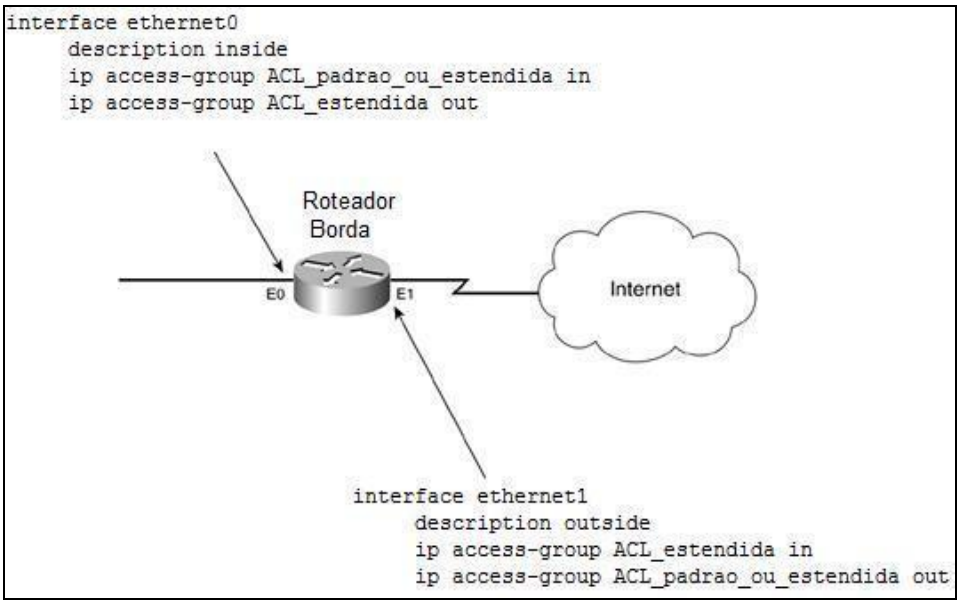

Figura 15 – Uso apropriado de ACLs CBAC.

Etapa 3 – Configuração de *Timeout*.

Opcionalmente, podemos mudar o *timeout* na tabela de estados, com os seguintes comandos:

```
Router(config)# ip inspect tcp synwait-time #_de_segundos
Router(config)# ip inspect tcp finwait-time #_de_segundos
Router(config)# ip inspect tcp idle-time #_de_segundos
Router(config)# ip inspect udp idle-time #_de_segundos
Router(config)# ip inspect dns-timeout #_de_segundos
```

Abaixo, em negrito, seguem os parâmetros *timeout* que podem ser configurados:

```
Router(config)#ip inspect ?
  alert-off       Disable alert
  audit-trail     Enable the logging of session information
(addresses and bytes)
  dns-timeout     Specify timeout for DNS
  max-incomplete  Specify maximum number of incomplete connections
before clamping
  name            Specify an inspection rule
  one-minute      Specify one-minute-sample watermarks for clamping
  tcp             Config timeout values for tcp connections
  udp             Config timeout values for udp flows

Router(config)#ip inspect tcp ?
  finwait-time  Specify timeout for TCP connections after a FIN
  idle-time     Specify idle timeout for tcp connections
  synwait-time  Specify timeout for TCP connections after a SYN and
no further data

Router(config)#ip inspect udp ?
  idle-time  Specify idle timeout for udp
```

```
Router(config)#ip inspect dns-timeout ?
  <1-2147483>  Timeout in seconds
```

O comando ***ip inspect tcp synwait-time*** especifica quanto tempo o Cisco IOS espera para que uma conexão TCP seja estabelecida (completar o *tree-way-handshake*). O padrão é 30 segundos. Se o *tree-way-handshake* não for completado até o limite desse tempo, ou seja, até finalizar o *timeout*, o Cisco IOS remove a entrada da tabela de estados e também a entrada criada dinamicamente na ACL (antes do recurso FAB, descrito anteriormente no item 5.3.3), e notifica ambas as partes que a conexão foi terminada.

O comando ***ip inspect tcp finwait-time*** especifica quanto tempo o Cisco IOS espera até remover uma entrada da tabela de estados quando a origem ou o destino iniciam o processo de desconexão de uma sessão TCP. O padrão é 5 segundos. Quando este processo é iniciado o Cisco IOS dá esse tempo aos dois dispositivos até que esta conexão seja removida da tabela de estados e da ACL (antes do recurso FAB).

O comando ***ip inspect tcp idle-time*** especifica quanto tempo o Cisco IOS mantém na tabela de estados uma conexão TCP ociosa. Uma conexão ociosa que estabeleceu uma conexão mas não tem tráfego fluindo nesta conexão. O padrão é 3600 segundos (1 hora). Quando este período expira (*timeout*), o IOS remove a entrada da tabela de estados e da ACL (antes do recurso FAB).

O comando ***ip inspect udp idle-time***, da mesma forma como acontece em uma conexão TCP, especifica quanto tempo o Cisco IOS mantém na tabela de estados uma conexão UDP ociosa. O padrão é 30 segundos. Depois que o período expira, o IOS remove a entrada UDP da tabela de estados e da ACL correspondente (antes do recurso FAB).

O comando ***ip inspect dns-timeout*** especifica quanto tempo o Cisco IOS mantém uma conexão de consulta DNS na tabela de estados. O padrão é 5 segundos. Quando o período expira o Cisco IOS remove esta consulta da tabela e da ACL correspondente (antes do recurso FAB). Este temporizador substitui o temporizador UDP ocioso. Este temporizador é usado para evitar *ip spoofing* das respostas DNS, fornecendo uma janela menor, dificultando o trabalho de um *Cracker* que utiliza as respostas DNS para redirecionar um dispositivo interno para um local errado.

Uma observação importante é que não existem comandos temporizadores para o tráfego ICMP. Existem outros métodos para esta configuração, como por exemplo o método conhecido como *inspection-by-inspection* (inspeção por inspeção), mas fogem do escopo deste trabalho.

Etapa 4 – *Port Application Mapping* (PAM).

CBAC usa *Port Application Mapping* (PAM) para determinar que tipo de inspeção deve ser feita em uma conexão. Por exemplo, o aplicativo padrão associado a porta 25 é o SMTP. Portanto, CBAC entende, por padrão, que o email é usado nesta conexão e, conseqüentemente, pode inspecionar esta conexão para os comando SMTP. Como outro exemplo, temos as conexões de controle FTP que é TCP na porta 21. Outra vez, CBAC entende que esta porta é usada pelo FTP e executa a inspeção apropriada nesta conexão.

PAM é usado para "remapear" portas ou associar portas adicionais com uma aplicação específica para que CBAC possa executar a inspeção apropriada nesta conexão. Como exemplo, podemos ter um servidor HTTP rodando na porta 8080. Por padrão, CBAC trata como uma conexão TCP normal. Para que esta conexão seja inspecionada pelo CBAC e seja tratada como uma conexão HTTP é preciso associar a porta 8080 com o HTTP. PAM é usado para mapear portas fora do padrão com aplicações. Pode ser usado para associar portas TCP e UDP à aplicações. PAM ainda permite associar porta a um determinado host para especificar uma aplicação. Por exemplo, pode existir somente um servidor HTTP rodando na porta 8080. PAM pode ser usado para associar somente esta porta a este servidor para que o CBAC inspecione a aplicação HTTP. Qualquer outra porta 8080 em qualquer outro dispositivo será tratado como uma conexão TCP normal. Com este recurso, reduz-se o número de inspeções que o CBAC tem que realizar, limitando somente àqueles dispositivos que usam portas fora do padrão.

O mapeamento de portas é colocado na tabela PAM, e o CBAC usa essa tabela para executar a inspeção adequada em uma conexão. São usados três tipos de entradas nesta tabela:

- ***System-defined entries*** – Estas são os números de portas conhecidas associadas as aplicações, como por exemplo, TCP porta 80 para HTTP. Estas entradas não podem ser deletadas ou mudadas. No entanto podemos substituir o *system-defined entries* para um host, como no exemplo dado anteriormente na aplicação HTTP usando a porta 8080. A Tabela 4 mostra entradas em uma tabela PAM.

Table 4 – PAM, Entradas System-Defined			
Aplicação	**Número da Porta**	**Aplicação**	**Número da Porta**
FTP	21	sql-net	1521
telnet	23	streamworks	1558
smtp	25	h323	1720
dns	53	netshow	1755
tftp	69	skinny	2000
http	80	mgcp	2427
sunrpc	111	sip	5060
msrpc	135	vdolive	7000
https	443	realmedia	7070
exec	512	cuseeme	7648
login	513	rstp	554 and 8559
shell	514		

- ***User-defined entries*** – estas são aplicações rodando em portas fora do padrão, como o servidor web rodando na porta 8080. Estas entradas podem ser criadas na tabela PAM para especificar um determinado número de porta. Também podem ser mapeados intervalos de portas para uma aplicação específica.
- ***Host-specific entries*** – Este é um subconjunto das *user-defined entries*, onde o mapeamento PAM mapeia somente conexões para um host específico, mas não todas as conexões usando o mesmo número de porta. Por exemplo, existir duas aplicações rodando na porta 8080, um servidor web e outra aplicação qualquer também na porta 8080. Com PAM, pode ser adicionada uma entrada na tabela PAM somente para o servidor web na porta 8080. Isto fará com que haja uma inspecione CBAC somente para aplicação HTTP na porta 8080, e uma inspeção TCP normal para outra aplicação na porta 8080.

Configurar o PAM é necessário somente se estiver executando aplicações em portas fora do padrão e se for desejado que haja inspeção CBAC nesta conexão. A configuração é simples:

```
Router(config)# ip port-map nome_aplicacao port num_porta [list acl_#]
```

Após configurar o mapeamento de portas, podem ser visto pelos comandos:

```
Router# show ip port-map
Router# show ip port-map nome_aplicacao
Router# show ip port-map port num_porta
```

Exemplo de uma saída parcial do comando ***show ip port-map***:

```
Router# show ip port-map
Default mapping: dns              port 53          system defined
Default mapping: tftp             port 69          system defined
Default mapping: https            port 443         system defined
Default mapping: realmedia        port 7070        system defined
Default mapping: ftp              port 21          system defined
Default mapping: telnet           port 23          system defined
!
<--saida omitida-->
```

O exemplo abaixo mostra 192.1.1.2 rodando um servidor web na porta 8080 e 192.1.1.3 também rodando um servidor web na porta 8090. Os primeiros dois comandos associam a inspeção HTTP na porta 8080 com 192.1.1.2 e dois últimos comandos associam a inspeção HTTP na porta 8090 com 192.1.1.3.

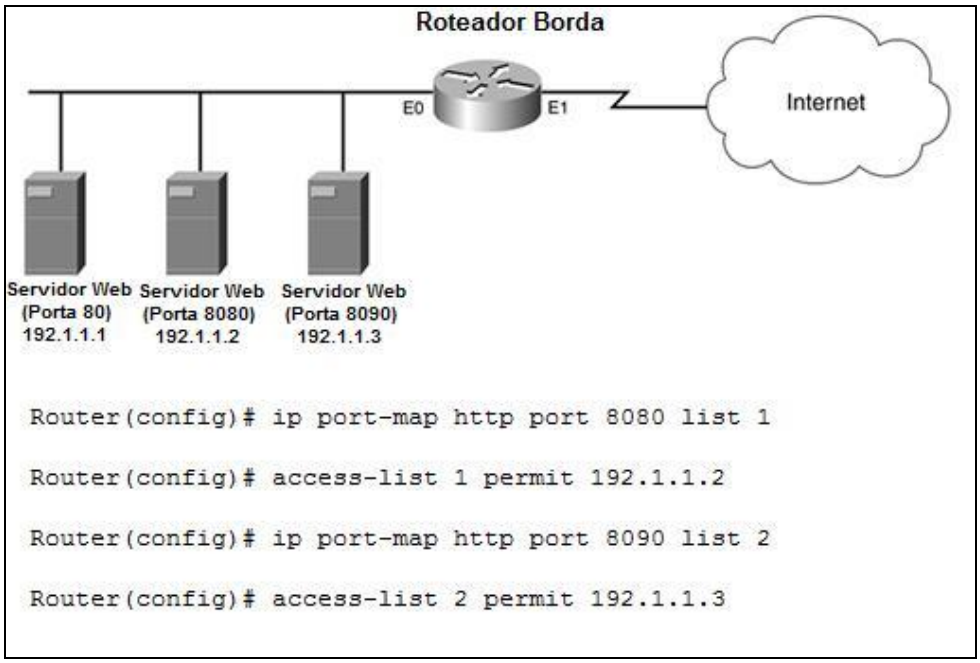

Figura 16 – Exemplo de configuração PAM.

Etapa 5 – Regras de Inspeção.

O ponto principal do CBAC são a regras de inspeção que são criadas. As regras de inspeção definem qual tráfego devem ser inspecionado. Por padrão não existem regras de inspeção predefinidas. As regras devem ser criadas manualmente. A sintaxe geral para regras de inspeção é a seguinte:

```
Router(config)# ip inspect name nome_da_regra protocol [alert {on | off}] [audit-trail {on | off}] [timeout seconds]
```

```
Router(config)#ip inspect ?
  alert-off       Disable alert
  audit-trail     Enable the logging of session information
(addresses and bytes)
  dns-timeout     Specify timeout for DNS
  max-incomplete  Specify maximum number of incomplete connections
before clamping
  name            Specify an inspection rule
  one-minute      Specify one-minute-sample watermarks for clamping
  tcp             Config timeout values for tcp connections
  udp             Config timeout values for udp flows

Router(config)#ip inspect name ?
  WORD  Name of inspection defined (16 characters max)

Router(config)#ip inspect name  REGRA1 ?
  http    HTTP Protocol
  icmp    ICMP Protocol
  tcp     Transmission Control Protocol
  telnet  Telnet
  udp     User Datagram Protocol

Router(config)#ip inspect name  REGRA1 {http| icmp| tcp| telnet| udp} ?
  alert        Turn on/off alert
  audit-trail  Turn on/off audit trail
  timeout      Specify the inactivity timeout time
  <cr>
```

As regras são agrupadas com um nome. Este nome é similar aos nomes ou números usados nas ACLs: as regras são associadas a um grupo particular. No exemplo abaixo, são mostradas as regras associadas à regra criada de nome “REGRA1”. Opcionalmente, podem ser ativados ou desativados os alertas e auditorias baseados por aplicação ou por protocolo pelos parâmetros *alert* e *audit-trail*, respectivamente. Se estes parâmetros forem omitidos, CBAC usa a configuração padrão: para o parâmetro *alert* o padrão é *on* e para o parâmetro *audit-trail* o padrão é *off*. Outro parâmetro opcional é o parâmetro *timeout*. Da mesma forma como nos parâmetros *alert* e *audit-trail*, se os valores não forem configurados é assumido o valor padrão. No exemplo a seguir, não foram definidos os valores, então, foram assumidos os valores padrão.

Exemplo:

```
Router#configure terminal
Router(config)#ip inspect name REGRA1 tcp
Router(config)#ip inspect name REGRA1 udp
Router(config)#ip inspect name REGRA1 icmp
Router(config)#ip inspect name REGRA1 telnet
Router(config)#ip inspect name REGRA1 http
Router(config)#exit
Router#sh ip inspect all
Inspection Rule Configuration
  Inspection name REGRA1
    tcp alert is on audit-trail is off timeout 3600
    udp alert is on audit-trail is off timeout 30
    icmp alert is on audit-trail is off timeout 10
    http alert is on audit-trail is off timeout 3600
    telnet alert is on audit-trail is off timeout 3600
```

Etapa 6 - Ativar Regras.

Depois de criar as regras, elas devem ser aplicadas no roteador. Como nas ACLs, elas são aplicadas em uma interface do roteador. O comando ***ip inspect*** é usado para ativar a regra na interface:

```
Router(config)# interface tipo_interface
Router(config-if)#  ip inspect nome_da_regra {in | out}
```

Etapa 7 – Testes da Configuração.

Após a configuração, temos três opções para monitorar o processo de inspeção e resolver possíveis problemas: comandos ***show***, comandos ***debug*** e alertas e auditorias.

A melhor maneira de testar CBAC é iniciando uma conexão que foi configurada para ser inspecionada, e então usar esses comandos para inspecionar o processo.

Comandos ***show***:

```
Router#sh ip inspect ?
  all         Inspection all available information
  config      Inspection configuration
  interfaces  Inspection interfaces
  name        Inspection name
  sessions    Inspection sessions
  statistics  Inspection statistics

Router# show ip inspect name REGRA2
show ip inspect name inspect_outbound
Inspection name REGRA2
  tcp alert is on audit-trail is off timeout 3600
  udp alert is on audit-trail is off timeout 30
```

Comandos ***debug***:

```
Router#debug ip inspect ?
  detailed         Inspection Detailed Debug Records
  events           Inspection events
  function-trace   Inspection function trace
  object-creation  Inspection Object Creations
  object-deletion  Inspection Object Deletions
  protocol         protocol-specific-debug
  timers           Inspection Timer related events
```

Alertas e Auditoria:

Alertas mostram mensagens na console do roteador relativas à operação do CBAC, como por exemplo, insuficiência de recursos do roteador, ataques DoS e outros. Por padrão, os alertas estão ativados. Para desabilitar os alertas usa-se o comando ***ip inspect alert-off***. Lembrando que é possível ativar ou desativar alertas por regra de inspeção. Seguem os parâmetros que podem ser usados com o comando ***ip inspect***:

```
Router(config)#ip inspect ?
alert-off       Disable alert
audit-trail     Enable the logging of session information (addresses
and bytes)
dns-timeout     Specify timeout for DNS
max-incomplete  Specify maximum number of incomplete connections
before clamping
name            Specify an inspection rule
one-minute      Specify one-minute-sample watermarks for clamping
tcp             Config timeout values for tcp connections
udp             Config timeout values for udp flows

Router(config)#ip inspect alert-off
```

5.3.6 **Conclusão**

Neste tópico foi mostrado o uso do CBAC com o uso de ACLs para implementar filtros e inspeção do tráfego. A Cisco recomenda o uso de CBAC sobre RACLs devido a facilidade na configuração, bem como seus recursos avançados, incluindo inspeção de aplicação. Com inspeção de aplicações, CBAC pode, por exemplo, monitorar conexões para limitar os comandos executados nestas conexões, e então evitar certos tipos de ataques DoS.

5.4 ACLs DINÂMICAS (LOCK AND KEY)

Uma questão que certamente administradores de rede têm que enfrentar é permitir usuários acessarem remotamente a sua rede, geralmente através da rede pública, pela Internet.

Em muitas situações são usadas Virtual Private Networks (VPNs)[16] para fornecer a conectividade. Entretanto, uma limitação que as VPNs têm é que, depois que usuários estão conectados através de uma conexão segura[17], têm liberdade total sobre os recursos internos. Pode ser aplicada uma ACL para restringir o tráfego, mas esta ACL é aplicada para todos os usuários que acessam um recurso e não somente a determinados usuários. ACLs normais, como padrão e estendidas, não fornecem esta funcionalidade, elas filtram somente nas camadas 3 e 4 e não podem fornecer acesso autorizado baseado na identificação de usuários.

Alguns mecanismos são necessários para autenticar usuários e restringir quais recursos eles podem acessar. Lock-and-Key ACLs, também chamadas de ACLs Dinâmicas, é uma solução que a Cisco apresenta para este tipo de problema.

Lock-and-Key ACLs foi introduzida no Cisco IOS 11.1. Esta foi a primeira solução para resolver o problema de autenticação de usuários ou dispositivos desconhecidos. Lock-and-Key ACLs usam ACL dinâmicas, parecidas com Context-Based Access Control (CBAC) e Reflexives ACLs (RACLs). Entretanto, CBAC e RACLs adicionam entradas baseadas na inspeção do tráfego, permitindo o tráfego de retorno para rede interna. Já uma Lock-and-Key ACL primeiramente requer autenticação do usuário através de Telnet ou SSH. Depois que o usuário é autenticado, entradas são ativadas na ACL estendida aplicada na interface. Estas entradas permanecem ativas por um período de tempo e depois expiram. Isto permite que usuários se autentiquem e acessem recursos internos, que normalmente seriam negados com aplicação somente de uma ACL estendida. Lock-and-Key ACLs são usadas basicamente em uma situação: quando é preciso restringir acesso à rede interna baseado na identificação dos usuários. Primeiro o usuário é autenticado, ou seja, fornece um *username* e/ou senha, e depois o acesso é permitido.

5.4.1 **Comparação entre ACLs Dinâmicas e ACLs Estáticas**

O processo de uma Lock-and-Key ACL é diferente de uma ACL estendida normal. Com uma ACL estendida todas as entradas são estáticas. Portanto, se for necessário permitir que um usuário específico acesse recursos internos na rede, será preciso criar uma entrada estática específica para garantir este acesso. Porém há um problema, esta entrada permanece ativa por

[16] Rede Particular Virtual (Virtual Private Network - **VPN**) é uma rede de comunicações privada normalmente utilizada por uma empresa, construída em cima de uma rede de comunicações pública (como por exemplo, a Internet). O tráfego de dados é levado pela rede pública utilizando protocolos padrão, não necessariamente seguros.

[17] **VPNs seguras** usam protocolos de criptografia por tunelamento que fornecem a confidencialidade, autenticação e integridade necessárias para garantir a privacidade das comunicações requeridas. Quando adequadamente implementados, estes protocolos podem assegurar comunicações seguras através de redes inseguras.

todo o período que a ACL estiver aplicada à interface e a interface estiver ativa (*up*). Obviamente, se o IP desse usuário não for conhecido, terá que ser permitido um grande intervalo de endereços, que pode comprometer a segurança ou ser contra as políticas aplicadas. Lock-and-Key ACLs são mais robustas que as ACLs estáticas por alguns motivos:

- Lock-and-Key ACLs autenticam usuários individuais, ACLs estáticas não;
- As entradas em ACL comuns são estáticas, e com isto é necessário que seja aberto um grande intervalo para permitir que usuários acessem recursos específicos, comprometendo a segurança. Com as Lock-and-Key ACLs entradas são criadas dinamicamente, baseadas na autenticação de usuários;
- ACLs estáticas são mais difíceis de gerenciar, especialmente quando é preciso controlar o acesso de usuários a recursos específicos, o que também pode causar processamento excessivo no roteador quando precisar processar muitas entradas de uma ACL.

5.4.2 **Funcionamento das Lock-and-Key ACLs**

Resumidamente, o processo inicia com a aplicação de uma ACL estendida para bloquear o tráfego externo. Usuários externos que querem estabelecer conexão com a rede interna serão negados pela ACL estendida até que estabeleçam uma conexão Telnet ou SSH para serem autenticados. Após a autenticação com sucesso, a sessão Telnet ou SSH é terminada e uma entrada ACL dinâmica é adicionada na ACL estendida aplicada anteriormente. Isto permite tráfego por um determinado período de tempo.

A Figura 17 exibe o processo para permitir que usuários acessem recursos internos, quando uma Lock-and-Key ACL está aplicada no roteador de borda. Logo abaixo é explicado o processo com mais detalhes.

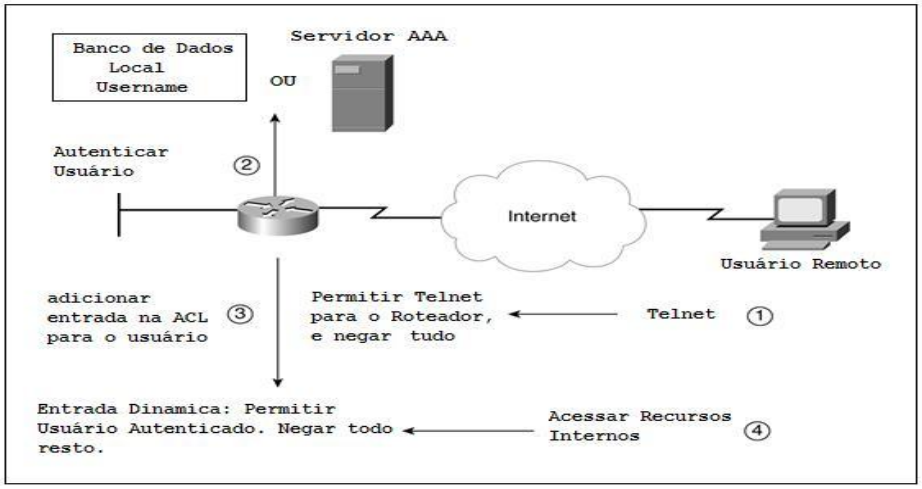

Figura 17 – Processo de Lock-and-Key ACLs.

A seguir são mostradas as etapas numeradas na Figura 17:

1 – Um usuário remoto abre uma conexão Telnet ou SSH para o roteador. A ACL estendida aplicada na interface externa do roteador deve permitir esta conexão. É solicitado um nome de usuário (*username*) e uma senha. Uma observação importante é que em acessos através da rede pública, é aconselhável usar SSH ao invés de Telnet para autenticação em conexões. SSH criptografa a sessão incluindo a senha enviada para o roteador, já com uso de Telnet, todo tráfego é enviado em texto claro, incluindo senhas.

2 – O roteador autentica a conexão. Três opções podem ser usadas para autenticação: nome de usuário no banco de dados local, um servidor AAA[18] usando RADIUS[19] ou TACACS+[20], ou usando comandos de senhas nas linhas VTY. Se a autenticação falhar, o *prompt* é mostrado novamente para que o usuário entre com as informações corretas. Se a autenticação for bem sucedida a conexão Telnet ou SSH é terminada. A função do Telnet ou SSH é somente para autenticação. Depois que o usuário for autenticado, a conexão Telnet ou SSH não é mais necessária, sendo então terminada.

3– Após o usuário ser autenticado com sucesso, o Cisco IOS adiciona uma entrada ACL dinâmica para garantir que o usuário acesse os recursos internos. Como será visto mais adiante, não podem ser criadas políticas por usuário. Ao invés disso, é definida uma política *lock-and-key* para todos os usuários, e esta única política é aplicada para todos os usuários autenticados.

4 – Neste ponto o usuário pode acessar os recursos internos que de outro modo seriam negados (através da entrada ACL estática). Se usuários não se autenticarem primeiro, eles terão acesso somente a recursos especificados na ACL estática aplicada na interface externa do roteador. Para acessar outros recursos internos os usuários devem primeiro se autenticar através de Telnet ou SSH. Em seguida, a entrada ACL dinâmica adicionada pelo roteador permite acesso a estes recursos.

---

[18] o termo protocolos **AAA** é uma referência aos protocolos relacionados com os procedimentos de *autentication, autorization e accounting* (autenticação, autorização e contabilidade*)*. A autenticação verifica a identidade digital do usuário de um sistema, a autorização garante que um usuário autenticado somente tenha acesso aos recursos autorizados e, por fim, a contabilidade refere-se a coleta de informações sobre o uso dos recursos de um sistema pelos seus usuários.

[19] Remote Authentication Dial In User Service (**RADIUS**) É um protocolo de rede que fornece (AAA). RADIUS foi desenvolvido pela Livingston Enterprises, Inc., em 1991 como um servidor de autenticação de acesso e do protocolo de accouting e, posteriormente, levados para a IETF normas. RADIUS é definido pela RFC 2865.

[20] Terminal Access Controller Access-Control System Plus (**TACACS+**) é um protocolo proprietário da Cisco que fornece controle de acesso para roteadores, acesso a servidores de rede e outros dispositivos de rede através de um ou mais servidores centralizados. TACACS+ fornece os serviços de *autentication, autorization e accounting*.

Uma observação importante quanto à segurança é que Lock-and-Key possibilita ataques do tipo *IP spoofing*. Depois que um usuário se autentica, abre um "buraco" temporário no firewall. Um *Cracker* pode usar o endereço IP de origem do usuário para explorar um ataque *spoofing* (geralmente, um ataque DoS). Para evitar este tipo de ataque é preciso considerar criptografia, como VPNs. A entrada ACL dinâmica temporária não é apagada automaticamente quando o usuário termina a sessão. Ela permanece na ACL até que o tempo limite seja alcançado ou até que o administrador apague a entrada manualmente.

5.4.3 **Configurando Lock-and-Key ACLs**

Lock-and-Key usa ACLs estendidas, na realidade uma entrada ACL dinâmica é inserida em uma ACL estendida, e esta entrada dinâmica cria uma entrada temporária na ACL para usuários autenticados. Portanto, é necessário um bom entendimento sobre a configuração e funcionamento de ACLs estendidas. Este tópico explicará as etapas para configuração de ACL dinâmicas. São três etapas básicas para confuguração:

1 – Criar a ACL estendida. No mínimo, esta ACL deve ter uma entrada para permitir acesso Telnet e/ou SSH para o roteador, bem como especificar onde será inserida a entrada dinâmica que será criada na autenticação de usuários, ou seja, uma referência para entrada dinâmica.

2– Definir a autenticação. Lock-and-Key suporta três métodos de autenticação: local (banco de dados local armazenado no roteador), um servidor externo AAA e através de senhas nas linhas VTY. Normalmente não são usados os comandos se senha nas linhas VTY porque todos os usuários usariam a mesma senha.

3 – Habilitar o método de autenticação lock-and-key. Isto é habilitado nas linhas VTY do roteador. Quando o método é habilitado o roteador cria uma entrada ACL dinâmica na ACL aplicada na interface que tem a referência lock-and-key.

Etapa 1 – Criar a ACL estendida.

A primeira coisa a fazer é criar uma ACL estendida para interface externa do roteador. Lock-and-key suporta ACLs estendidas numeradas ou nomeadas. Esta ACL deve permitir acesso Telnet ou SSH para um endereço IP no roteador. Normalmente este é o endereço configurado na interface externa. Depois disso, introduz-se as entradas da ACL lock-and-key na ACL estendida. Estas entradas definem quais recursos internos serão permitidos aos usuários após se autenticarem.

Os seguintes comandos criam as entradas ACL dinâmicas:

```
Router(config)# access-list ACL_# dynamic nome_ACL [timeout minutos]
{deny | permit} protocolo_IP IP_origem mascara_origem IP_destino
mascara_destino [precedence precedence] [tos TOS] [established]
[log]
Ou
Router(config)# ip access-list extended nome_ACL
Router(config-ext-nacl)# dynamic nome_ACL [timeout minutos]
{deny | permit} protocolo_IP IP_origem mascara_origem IP_destino
mascara_destino [precedence precedence] [tos TOS] [established]
[log]
```

O parâmetro *dynamic* nessas duas ACLs especifica o nome da ACL dinâmica que será usada. Este nome deve ser único entre todas as ACLs criadas no roteador. O parâmetro *timeout* é opcional. Ele especifica o tempo para a entrada dinâmica. O *timeout* pode variar de 1 a 9999 minutos. Dois *timeouts* opcionais, podem ser associados a uma entrada ACL dinâmica: *absolute* e *idle*. O tempo *absolute* é especificado na entrada ACL dinâmica. O *absolute timeout* é o tempo em minutos, especificado para que uma conexão seja finalizada. É habilitado nas linhas *vty* pelo comando ***absolute-timeout***. Se este tempo não for especificado, o padrão é nunca expirar o tempo desta entrada. O *idle timeout* é o tempo de espera para que uma conexão ociosa seja finalizada. É especificado junto com o comando ***autocommand access-enable [timeout valor]***, que é usado para habilitar autenticação lock-and-key nas linhas VTY. Da mesma forma que o *absolute timeout*, se este tempo não for especificado, o padrão é que esta entrada nunca expire e permaneça na ACL estendida. Pode ser configurado um dos dois *timeouts* ou ambos, porém se os dois forem configurados, o valor do *idle timeout* deve ser menor que o valor do *absolute timeout*.

Após a configuração do parâmetro opcional *timeout*, deve ser especificado que tipo de tráfego será pemitido para os usuários. Normalmente, não se sabe o enderço IP do usuário externo (o endereço de origem), portanto, é usado o parâmetro *any*. Existe uma opção no comando ***autocommand*** que permite que o Cisco IOS substitua o parâmetro *any* pelo endereço IP de origem do usuário autenticado. Este parâmetro será mostrado na terceira etapa (Habilitar o método de autenticação lock-and-key).

Após criar a ACL estendida com as permissões Telnet e/ou SSH e a entrada dinâmica lock-and-key, ela deve ser aplicada na interface externa do roteador com o comando ***ip access-group***.

```
Router(config)#interface fastEthernet 0/0
Router(config-if)#ip access-group ACL_TESTE in
```

O comando ***access-list dynamic-extended***, mostrado abaixo, extende o valor do timeout para entradas lock-and-key por seis minutos.

```
Router(config)# access-list dynamic-extended
```

Este é um comando opcional, normalmente usado quando existe um trabalho em execução e está próximo do *absolute timeout*, mas o trabalho não vai terminar a tempo. Configurando este comando no roteador, o valor *timeout* é extendido por seis minutos.

Etapa 2 – Definir o método de autenticação.

Aqui serão apresentados os três métodos de autenticação descritos anteriomente: senha nas linhas VTY, banco de dados local e servidor AAA. Seguem abaixo as configurações:

- Usando senha nas linhas VTY:

```
Router(config)# line vty 0 4
Router(config-line)# login
Router(config-line)# password senha
```

Outra vez observando que, não é recomendado usar o método de senha nas linhas VTY para autenticação de usuários, porque todos os usuários usariam a mesma senha.

- Usando o banco de dados local:

```
Router(config)# username nome secret senha
Router(config)# line vty 0 4
Router(config-line)# login local
```

O comando ***show running-config*** exibe os usuários cadastrados localmente.

```
Router#sh running-config
<saída omitida>
!
!
username MOT2006 secret 5 $1$mERr$OAZJyntnash.EflFFzcMJ1
username Moacyr secret 5 $1$mERr$SI6kKbhlkuiS3Lv8zc1kp1
username Vicente secret 5 $1$mERr$H7PDxl7VYMqaD3id4jJVK/
!
!
<saída omitida>
```

- Para configurar AAA com TACACS, use a seguinte configuração:

```
Router(config)# aaa new-model
Router(config)# tacacs-server host endereco_IP
Router(config)# tacacs-server key senha
Router(config)# aaa authentication login authentication_nome group tacacs+
Router(config)# line vty 0 4
Router(config-line)# login authentication authentication_nome
```

Etapa 3 – Habilitar o método de autenticação lock-and-key.

A terceira e última etapa é habilitar a autenticação *lock-and-key* nas linhas VTY do roteador. Segue abaixo a configuração:

```
Router(config)# line vty 0 4
Router(config-line)# autocommand access-enable host [timeout minutos]
```

O comando ***autocommand access-enable*** especifica a autenticação *lock-and-key*. Quando um usuário é autenticado com sucesso, uma entrada ACL temporária é inserida no espaço reservado pelo parâmetro *dynamic* na ACL estendida aplicada na interface onde o usuário está tentando a conexão. Sem o comando ***autocommand access-enable*** o roteador não criará a entrada ACL temporária.

O parâmetro *host* é opcional. Ao especificar este parâmetro o Cisco IOS substitui o parâmetro *any* na entrada da ACL dinâmica pelo endereço IP do usuário. Se a ACL estendida for aplicada na entrada (*inbound*) da interface externa do roteador, o parâmetro *any* de origem é alterado pelo endereço IP do usuário. Se aplicada na saída (*outbound*) da interface externa, o parâmetro *any* de destino é substituído.

5.4.4 **Conclusão**

Neste tópico foi apresentado o básico das ACLs lock-and-key, ou ACLs dinâmicas, que foi introduzida no Cisco IOS 11.1 para possibilitar autenticação de usuários e garantir que acessem recursos internos na rede. A ACL dinâmica é dependente do acesso via Telnet ou SSH, autenticação local ou remota e aplicação de ACLs estendidas. A configuração de uma ACL dinâmica inicia com a aplicação de uma ACL estentida para filtrar o tráfego externo através do roteador. Usuários que desejam acessar recursos internos serão bloqueados pela ACL estendida até que iniciem uma sessão Telnet para o roteador e sejam autenticados. Após a autenticação bem sucessida, a sessão Telnet é finalizada e uma entrada dinâmica é adicionada na ACL estendida já aplicada anteriormente na interface do roteador de borda. Então, é permitido o tráfego aos recursos internos, que podem ser configurados para permanecerem disponíveis, ou acessíveis, por somente um período de tempo predefinido, configurado pelos parâmetros opcionais *idle timeout* e *absolute timeout*. ACL dinâmica é uma alternativa de melhorar o acesso a rede, porém deve ser configurada com atenção, pois durante o processo é aberta uma "passagem" temporária para garantir o acesso dos usuários autenticados. Dessa forma, esta "passagem" também pode ser usada por *Crackers* para atacar a rede.

5.5 TIME-BASED ACLs

Às vezes administradores precisam aplicar filtros na rede baseados em períodos de tempo. Serviços ou recusos podem ser permitidos ou negados durante um período de tempo, que pode ser apenas algumas horas de um dia, durante dias inteiros ou uma combinação de horários e dias determinados. Por exemplo, um funcionário externo de uma empresa qualquer poderá estabelecer uma sessão Telnet para a rede interna somente durante o horário de expediente (podemos supor que seja de Segunda a Sexta entre 08:00 às 17:00h), sendo negada qualquer tentativa fora deste horário.

A Cisco introduziu no Cisco IOS 12.0.1.T a Time-Based ACL para fornecer controle de acesso baseado no tempo, e é suportado por todas as plataformas Cisco.

Um intervalo de tempo (*time range*) é criado para especificar horários do dia ou da semana em que será permitido ou negado determinado tipo de tráfego durante este período. Este período de tempo é identificado por um nome e depois referenciado em um ACL. Entretanto, existem restrições de tempo nesta função do roteador. O *time range* criado depende do relógio do sistema do roteador. Para que o *time range* funcione corretamente é necessária uma fonte confiável. O relógio do roteador pode ser usado, mas esta função funciona melhor com a sincronização do Network Time Protocol (NTP)[21].

5.5.1 Configurando Time-Based ACLs

A configuração é simples, e requer somente duas estapas: definir o *time range* e referenciar o *time range* a uma ACL estendida.

Para definir um *time range* deve ser usado o comando ***time-range*** no modo de configuração global:

```
Router(config)# time-range nome-time-range
Router(config-time-range)# absolute [start hora data] [end hora data]
```

e/ou

```
Router(config-time-range)# periodic dia-da-semana hh:mm to
[dia-da-semana] hh:mm
```

A seguir será criado um *time range* para o exemplo citado anteriormente, onde será permitido o acesso Telnet somente no horário de expediente (de Segunda a Sexta entre 08:00 e 17:00h).

---

[21] **O Network Time Protocol (NTP)** é um protocolo para sincronização de relógios em sistemas de computadores sobre comutação de pacotes, em redes de latência variável. NTP usa UDP na porta 123 como sua camada de transporte. Foi criado particularmente para para lidar com o problema de latência variável, usando *jitter buffer*. Os detalhes operacionais do NTP estão especificados em: RFC 778, RFC 891, RFC 956, and RFC 1305.

A primeira etapa é definir o *time range:*

```
Router(config)# time-range TELNET
Router(config-time-range)# periodic weekdays 08:00 to 17:00
Router(config-time-range)# exit
```

A segunda etapa é referenciar os parâmetros *permit ou deny* em uma ACL estendida:

```
Router(config)# ip access-list extended PERMITE_TELNET
Router(config-ext-nacl)# permit tcp any any eq 23 time-range TELNET
Router(config-ext-nacl)# permit tcp any any
Router(config-ext-nacl)# exit
```

Neste exemplo foi criado um *time range* de nome TELNET que especifica os dias da semana (*weekdays* - de segunda a sexta) no horário entre 08:00 e 17:00. Depois, o time range TELNET foi associado à ACL estendida, na qual foi criada uma regra para permitir o acesso Telnet (porta 23) somente nos dias e horários definidos pelo *time range* TELNET. Nos próximos tópicos serão apresentados os parâmetros disponíveis para configuração do *time range*.

5.5.1.1 Comando *Time-Range*

O parâmetro *time-range* é identificado por um nome, no qual é feita a referência na ACL estendida. Este nome não pode ter espaços, apóstrofos ou aspas, e deve começar por um caractere alfabético. Em uma única ACL estendida pode ocorrer múltiplos *time ranges.*

Após o comando ***time-range***, é usado o comando ***periodic***, o comando ***absolute***, ou alguma combinação deles para definir quando o ***time range*** estará ativo. Múltiplos comandos ***periodic*** são permitidos em um ***time range***, mas somente um comando ***absolute*** é permitido. Se um ***time range*** tiver configurados ambos os valores para os comandos ***absolute*** e ***periodic***, os valores do comando ***periodic*** só serão avaliados após serem alcançados os valores do comando ***absolute start***, e não serão mais avaliados após o ***absolute end*** ser alcançado. Então, serão ignorados de novo até que os valores ***absolute start*** sejam alcançados novamente. Ou seja, os valores do comando ***absolute*** têm prioridade sobre os valores do comando ***periodic***.

5.5.1.2 Parâmetro *Absolute*

```
Router(config-time-range)# absolute [start hora data] [end hora data]
```

`[start hora data]` – parâmetro opcional no qual os parâmetros *permit* ou *deny* estão associados para dar início ao processo de filtragem em uma ACL estendida. A hora é expressa no padrão 24 horas, na forma hh:mm. Por exemplo, 10:00 é o mesmo que 10:00 am

(padrão americano), e 22:00 é o mesmo que 10:00 pm (no padrão americano). A data é expressa no formato *day month year* (dia mês ano). O início mínimo é *00:00 1 January 1993*. Se não forem especificadas a hora e data de início, os parâmetros *permit* ou *deny* serão avaliados imediatamente.

`[`**`end`** `hora data]` – parâmetro opcional no qual os parâmetros *permit* ou *deny* estão associados para dar fim ao processo de filtragem em uma ACL estendida. O formato para hora e data é idêntico ao parâmetro *start*, descrito acima. Este parâmetro, obviamente, deve ser colocado após o parâmetro *start.* O tempo máximo final é *23:59 31 December 2035*. Se não forem especificadas a hora e data, os parâmetros *permit ou deny* serão avaliados infinitamente.

5.5.1.3 Parâmetro *Periodic*

```
Router(config-time-range)# periodic dia-da-semana hh:mm to
[dia-da-semana] hh:mm
```

`[dia-da-semana]` – a primeira entrada deste argumento é o dia de início ou os dias que serão associados para o filtro na ACL. A segunda entrada é o dia ou os dias que encerrarão os parâmetros de teste na ACL. Este argumento pode ser somente um dia ou uma combinação de dias: *Monday* (Segunda), *Tuesday* (Terça), *Wednesday* (Quarta), *Thursday* (Quinta), *Friday* (Sexta), *Saturday* (Sábado) e *Sunday* (Domingo).

Outros valores possíveis são:

***daily* –** Segunda a Domingo *(Monday - Sunday)*

***weekdays* -** Dias da Semana, de Segunda a Sexta *(Monday – Friday)*

***weekend* -** Finais de Semana, Sábado e Domingo (*Saturday e Sunday)*

Se o dia final for igual ao dia de início ele pode ser omitido.

`[hh:mm]` – a primeira entrada deste argumento é hora de início associado aos parâmetros ACL. A segunda entrada é o horário de encerramento, também associado aos parâmetros *permit* e *deny*. Igualmente ao parâmetro *absolute*, a hora e minutos são expressos no padrão 24 horas.

5.5.2 **Verificando a Atividade Time-Based ACL**

É possível verificar se uma *Time-Based ACL* está ativa ou inativa usando o comando ***show ip access-lists*** *[nome_ACL]*. Neste exemplo, será mostrado o comportamento para a *Time-Based ACL* PERMITE_TELNET, criada no exemplo do tópico 5.5.1 – configurando *Time-Based ACLs*. Se a *Time-Based ACL* for verificada durante o expediente (de Segunda a

Sexta entre 08:00 e 17:00), a ACL será mostrada como *active* (ativa), caso contrário será mostrada como *inactive* (inativa):

```
Router#show clock
1:46:58.917 GMT qua abr 7 2010
Router#show ip access-lists PERMITE_TELNET
Extended IP access list PERMITE_TELNET
    permit tcp any any eq telnet time-range TELNET (inactive)
    permit tcp any any
Router#
```

Agora foi alterado o horário para 10:05:36 (dentro do horário permitido), e a ACL foi ativada:

```
Router#show clock
10:05:36.465 GMT qua abr 7 2010
Router#show ip access-lists PERMITE_TELNET
Extended IP access list PERMITE_TELNET
    permit tcp any any eq telnet time-range TELNET (active)
    permit tcp any any
Router#
```

5.5.3 **Conclusão**

Time-Based ACLs são muito úteis quando é necessária a filtragem do tráfego, permitindo ou negando aplicações ou recursos, baseados em determinados intervalos de tempo. É necessário que a informação de tempo seja passada de fonte confiável para que a ACL funcione da forma desejada ou poderão ocorrer situações inesperadas. Por isso é aconselhável o sincronismo do protocolo NTP ao uso do relógio do sistema do roteador. A criação e aplicação da ACL estendida é um ponto importante, como nas demais ACLs, deve-se pensar com cuidado no que será permitido ou negado.

6 CONCLUSÃO

Administradores de redes vivem constantemente em uma “batalha” para garantir o perfeito funcionamento da rede, bem como garantir que os objetivos definidos nas políticas de negócios e segurança sejam alcançados. Para isto pode-se usar como complemento os recursos das ACLs - *Access Control Lists* (Listas de Controle de Acesso) disponíveis no Cisco IOS instalado em roteadores e switches *multilayer* (também operam na camada 3) de fabricação Cisco®.

Desde 1993 muitos administradores de redes usam dois tipos de ACLs: ACLs padrão e extendida. As ACLs padrão aplicam filtros baseadas somente no endereço IP de origem (*source IP address*). As ACLs estendidas podem aplicar filtros baseados nos endereços IP de origem e destino (*source* e *destination IP address*), protocolos, números de portas, *flags* TCP e mensagens ICMP, porém há limitações. Deve-se ter um bom ententimento quanto à utilização básica de ACLs: de que forma usá-las, como criá-las, e onde e como aplicá-las.

É também importante identificar as limitações de cada ACLs. Esses dois tipos de ACLs, especialmente as ACLs estendidas, podem ser ótimos recursos para filtragem de pacotes e aplicar mais um recurso de segurança na rede. Porém, não filtram todo tipo de tráfego, não podem impedir ou prevenir todos os tipos de ataques, ou permitir que somente usuários autorizados tenham acesso a recursos internos.

O uso básico de ACLs depende do filtro ou nível de proteção desejado. Entretanto, se for necessária a utilização de soluções mais robustas, como classificação do tráfego para implementar QoS, restrição de acesso (Telnet e SSH), filtros na camada de aplicação, filtros baseados em intervalos de tempo e autenticação de usuários nas conexões, pode-se combinar o uso das ACLs padrão e estendidas a outros recursos, como as ACLs mostradas neste trabalho: Turbo ACLs, ACLs Reflexivas, *Context-Based* ACLs, *Time-Based* ACLs e *Lock-and-Key* ACLs.

As ACLs, sejam básicas ou avançadas, são de extrema importância em uma rede. Sem uma lista de acesso todos os pacotes fluem livremente por toda a rede, o que pode não ser muito interessante para os administradores e para o bom funcionameto da rede. Em questões de segurança ACLs não são melhores ou piores que *Firewalls* e nem os substituem, apenas complementam e fornecem mais um nível na segurança. Como por exemplo as ACLs *stateful* que mantêm o estado das conexões, permitindo e verificando o tráfego que retorna para a rede interna. Outra característica importante das ACLs *stateful* é não deixar aberto o acesso externo após uma conexão ser finalizada, ou seja, diminui as possibilidades de ocorrer um ataque externo.

Atualmente um dos grandes desafios na administração de redes é manter um ambiente seguro, robusto e estabelecer um controle de tráfego de pacotes de acordo com as políticas adotadas, que nem sempre é possível somente com a utilização de Firewalls e ACLs básicas, ou às vezes, um ou outro, como acontece em muitos casos. O objetivo deste trabalho foi mostrar a importância das ACLs, bem como suas funcionalidades, criação e aplicação, para alcançar os objetivos propostos e vencer os desafios, que crescem a cada dia. Por isso a necessidade de utilizarmos também as ACLs avançadas em conjuntos com os demais recursos.

7 REFERÊNCIAS

[1] BLOG CCNA – Cisco Certified, Disponível em: <http://blog.ccna.com.br>. Acesso em: fev. 2010.

[2] CISCO.COM – Cisco Systems, Disponível em: <http://www.cisco.com>. Acesso em: mar. 2010.

[3] DOOLEY, Kevin; BROWN, Ian. *Cisco Cookbook*, 2nd Ed. O'Reilly, 2006.

[4] GLOBALKNOWLEDGE.COM – Global Knowledge, Disponível em: <http://www.globalknowledge.com>. Acesso em: mar. 2010.

[5] SADAYAO, Jeff. Cisco *IOS Access List*. First Ed. O'Reilly, 2001.

[6] TECHREPUBLIC – TechRepublic Blogs, Disponível em: <http://blogs.techrepublic.com.com/>. Acesso em: fev. 2010.

[7] WIKIPEDIA.ORG – Wikipédia, a enciclopédia livre, Disponível em: <http://pt.wikipedia.org/wiki/>. Acesso em: mar. 2010.

8 ANEXO

Flag TCP	**Representação do flag**	**Significado do flag**
SYN	"S"	Esta é uma requisição de estabelecimento de sessão que é a primeira parte de qualquer conexão TCP
ACK	“ack”	Este flag normalmente é usado para acusar o recebimento de dados ao emissor. Ele pode ser visto em conjunto (ou concatenação) com outros flags
FIN	“F”	Este flag indica a intenção do emissor de terminar elegantemente a conexão entre o host emissor e o host receptor.
RESET	“R”	Este flag indica a intenção do emissor de abortar imediatamente a conexão existente com o host receptor
PUSH	“P”	Este flag envia imediatamente dados ao host emissor para o software aplicativo no host receptor. Não há espera alguma para o buffer preencher. Nesse caso, o foco está na sensibilidade e não na eficiência da largura de banda. Para muitas aplicações interativas, como a Telnet, a principal preocupação é o menor tempo de resposta, que o flag PUSH tenta sinalizar
URGENT	“urg”	Este flag indica que dados 'urgentes' devem ter prioridade sobre outros dados. Um exemplo disso é pressionar Ctrl + C para abortar um download FTP.
Marcador de lugar	“.”	Se a conexão não tiver um flag SYN, FIN, RESET ou PUSH definido, um marcador de lugar (um ponto) será encontrado após a porta de destino.