UNIVERSIDADE DE CAXIAS DO SUL
DEPARTAMENTO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM GERÊNCIA E SEGURANÇA DE REDES DE COMPUTADORES

JERONIMO CLEBERSON ZUCCO

Hardening Linux Usando Controle de Acesso Mandatório

Prof. Ms. Alex Fábio Pellin
Orientador

Caxias do Sul, fevereiro de 2008

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Zucco, Jeronimo Cleberson

Hardening Linux Usando Controle de Acesso Mandatório / Jeronimo Cleberson Zucco. – Caxias do Sul: Universidade de Caxias do Sul, 2008.

65 f.: il.

Dissertação (pós-graduação) – Universidade de Caxias do Sul. Pós-Graduação em Gerência e Segurança de Redes de Computadores, Caxias do Sul, BR–RS, 2008. Orientador: Alex Fábio Pellin.

1. SELinux. 2. Hardening. 3. Linux. 4. Segurança. 5. MAC. I. Pellin, Alex Fábio. II. Título.

UNIVERSIDADE DE CAXIAS DO SUL

“There are no bugs,
just undocumented features.”
— ANONYMOUS SECURITY QUOTE

AGRADECIMENTOS

Agradeço a minha famíla, a minha namorada e a Deus.

Agradeço também a todos os desenvolvedores de software livre, por compartilhar o seu conhecimento.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ABNT Associação Brasileira de Normas Técnicas

API *Application Programming Interface*

AVC *Access Vector Cache*

BIND Berkeley *Internet Name Domain*

BLP Bell La-Padula *Security Model*

BSD Berkeley *Software Distribution*

CERT *Computer Emergency Response Team*

CERT.br *Computer Emergency Response Team* Brasil

CGI *Common Gateway Interface*

CIFS *Common Internet File System*

CSIRT *Computer Security Incident Response Team*

CVE *Common Vulnerabilities and Exposures*

DAC *Discretionary Access Control*

DNS *Domain Name System*

DOD *Department of Defense*

EAL *Evaluation Assurance Level*

FTP *File Transfer Protocol*

GUI *Graphical User Interface*

HTTP *Hypertext Transfer Protocol*

IDS *Intruder Detection System*

I/O *Input/Output*

IP *Internet Protocol*

IPC *Inter-process communication*

IPv6 *Internet Protocol version 6*

IPSEC *IP Security*

ITSEC *Information Technology Security Evaluation Criteria*

LSM *Linux Security Module*

LSPP *Labeled Security Protection Profile*

LTP *Linux Test Project*

MAC *Mandatory Access Control*

MCS *Multi-Category Security*

MD5 *Message-Digest Algorithm 5*

MLS *Multilevel Security*

NFS *Network File System*

NSA *National Security Agency*

NTP *Network Time Protocol*

OSI *Open Systems Interconnection*

PGP *Pretty Good Privacy*

PHP *PHP: Hypertext Preprocessor*

PID *Process Identification*

RBAC *Role-based access control*

SCC *Secure Computing Corporation*

SID *Security Identifier*

SL *Security Level*

SSH *Secure Shell*

SSI *Server Side Includes*

TCP *Transmission Control Protocol*

TCSEC *Trusted Computer System Evaluation Criteria*

TE *Type Enforcement*

UDP *User Datagram Protocol*

UFO *Unidentified Flying Object*

US *United States*

VFS *Virtual File System*

XML *Extensible Markup Language*

LISTA DE FIGURAS

RESUMO

Em computação, *hardening* é o processo de customizar um sistema em busca de maior segurança pró-ativa, para que ele se torne o mais resistente possível à ataques de *crackers*. Isso tipicamente inclui remoção dos usuários que não serão usados e da desativação ou remoção dos serviços e programas desnecessários. Esse trabalho irá abordar o uso do **SELinux** (*Security-Enhanced Linux*) para a realização dessa difícil tarefa de manter o sistema seguro e minimizar os efeitos desses ataques.

Palavras-chave: SELinux, Hardening, Linux, Segurança, MAC.

Hardening Linux Using Mandatory Access Controls

ABSTRACT

In computing, hardening is the process of customizing a system in search of greater pro-active security, to perform the most possible resistant to cracker's attacks. This typically includes removal of users who will not be used and disabling or removal unnecessary services and programs. This work will address the use of SELinux (Security-Enhanced Linux) to do the hard task of keeping the system safe and minimize the effects of those attacks.

Keywords: SELinux, Hardening, Linux, Security, MAC.

1 INTRODUÇÃO

É de conhecimento público que o número e nível de complexidade de ataques através da internet vem crescendo rapidamente nos últimos anos e, conforme a figura 1.1, com os dados dos ataques registrados pelo CERT.br (Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil), pode-se afirmar que, apesar de uma tendência de pequena queda em 2007 em relação a 2006, os índices devem se manter em níveis elevadíssimos, 200% maiores que a média dos 3 anos anteriores.

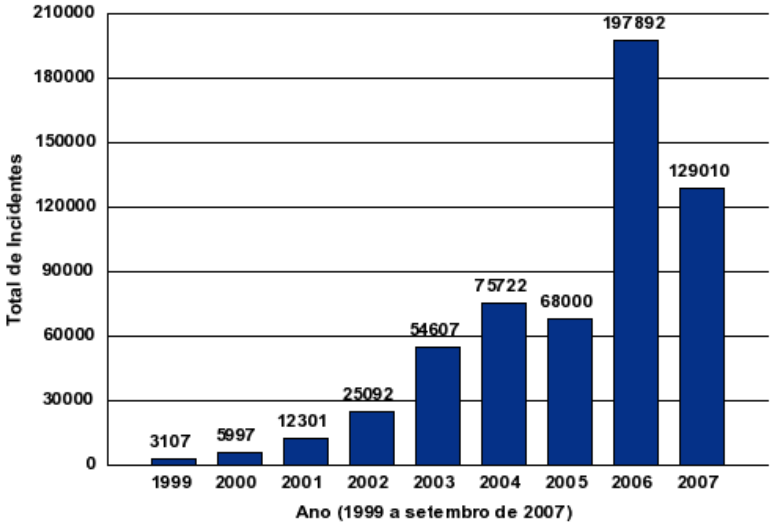

Figura 1.1: Total de Incidentes Reportados ao CERT por Ano

Nos gráficos das figuras 1.1 e 1.2 percebe-se o grande crescimento das notificações registradas em 2006, e das que foram feitas até setembro de 2007. Essa é uma maneira indireta de medir o crescimento do nível e número de ataques a serviços web, dado que esses números dependem da notificação dos administradores de sistemas para que os CERT's (*Community Emergency Response Teams*) contabilizem esses dados. Além disso, nem todos os ataques são originados a partir da internet, a grande maioria das vezes os ataques se utilizam da rede interna, seja por falha de segurança ou controle, como acesso a rede de máquinas não autorizadas, computadores infectados por malwares, os ditos zumbis, que funcionam como trojans.

Muitos administradores deixam de notificar incidentes de segurança por receio da perda de imagem da empresa onde trabalham ou até mesmo para não expor os problemas aos seus superiores, por medo de perder o emprego. Mas a maior parte dos casos de ataques não notificados se dá devido ao desconhecimento desse fato pelos administradores.

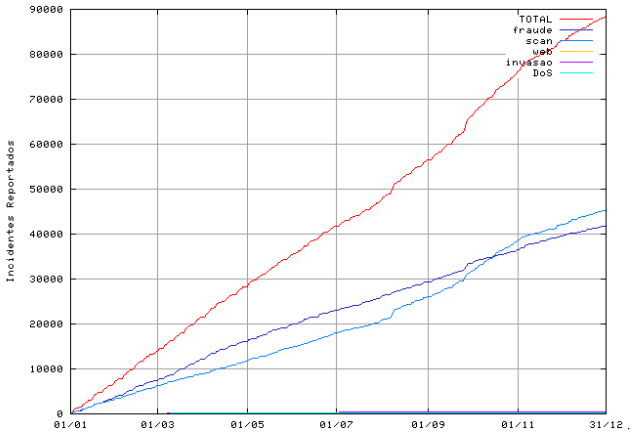

Figura 1.2: Incidentes Reportados ao CERT.br - Janeiro a Dezembro de 2006

A maioria dos administradores de redes de computadores não tomam conhecimento dos incidentes seja por falta de conhecimento dele próprio na hora que não implementa um firewall corretamente, ou não analisa logs, má configuração de serviços, etc, ou falta de recursos humanos e dispositivos que permitiriam esse tipo de análise, como IDS's (*Intruder Detection System*), analisadores de logs, CSIRT's (*Computer Security Incident Response Team*), etc.

Um fator que contribui para esse crescimento dos níveis de ataques é aumento exponencial da complexidade do software, conforme notado por (LOSCOCCO 1998). Como a inteligência humana é finita, desenvolvedores de software cometem erros ou omitem verificações de segurança durante a implementação de sistemas. Raramente passa-se um dia sem que algum vírus ou vulnerabilidade de alguma aplicação seja anunciada. Além disso, com a crescente necessidade de mão de obra para desenvolvimento de sistemas, muitos programadores não estão suficientemente preparados para desenvolver sistemas e programas de forma segura. Isso se dá por falta de experiência em determinada linguagem ou até mesmo pelo fato da linguagem ser muito permissiva para desenvolvedores como muitos softwares desenvolvidos na linguagem PHP, conforme pode ser observado em (LEMOS 2006) e (SHIFLETT 2005), por exemplo. Esse fato se tornou mais notório para aplicações web, onde uma única falha de segurança pode comprometer todo um sistema e até expor dados privados.

Outro fator que influi diretamente na quantidade de ataques é a popularização da internet e dos serviços por ela prestados. Um ataque direcionado a apenas uma máquina pode comprometer toda a rede de computadores na qual ela faz parte. Atacantes podem usar essas máquinas comprometidas para enviar outros ataques para outras máquinas não expondo a sua verdadeira identidade.

Outros fatores que devem ser citados:

- Contéudos ativos: documentos doc ou pdf, por exemplo, que são interpretados por um programa que podem ter falhas de segurança;
- Código móvel: códigos que são executados na máquina cliente, como javascript, Microsoft ActiveX, etc;

- Escalação de privilégios: o atacante pode usar privilégios concedidos ao serviço ou programa disponibilizado para executar um ataque para conseguir privilégios maiores na máquina atacada, isso quando o serviço não está rodando como o usuário administrador, o que ocorre em muitos casos;

- Ciclo de atualizações e *Zero Days*: o principal problema nesse caso é o fato que o lançamento de *patches* (correções de código) são em sua grande maioria reativos, e não pró-ativos. Entre o período da descoberta de uma vulnerabilidade em um sistema até o lançamento de uma correção para o problema, existe um período de exposição do usuário ao ataque (chamado de janela de vulnerabilidade). Além disso, após a disponibilização, o usuário deve adquirir, autenticar (via verificação PGP ou MD5, etc), testar e instalar a correção, além do fato que a própria correção pode expor o usuário a mais problemas de segurança.

1.1 Modelos de Controle de Acesso: DAC e MAC

O modelo mais conhecido de controle de acesso atualmente é modelo DAC (*Discretionary Access Control*), porém ele possui alguns pontos fracos que são intrísecos a sua natureza. Para superar esses pontos fracos, a comunidade de segurança desenvolveu mecanismos MAC (*Mandatory Access Control*). Suas principais diferenças serão vistas a seguir.

1.1.1 *Discretionary Access Control* - DAC

A evolução do controle de acessos, que em sistemas operacionais mais antigos era nenhuma, permitiu o surgimento de várias abordagens para resolver os problemas de segurança de acesso não autorizado. Uma dessas abordagens que surgiu foi a DAC (BELL 1975) (*Discretionary Access Control*). Controle de acesso discricionário é um tipo de controle de acesso definido no *Trusted Computer System Evaluation Criteria*, mais conhecido como *Orange Book* (NSA 1985), que depois foi substituído em 2005 pela norma ISO/IEC 15408:2005, também chamado de *Common Criteria* (ISO 2005).

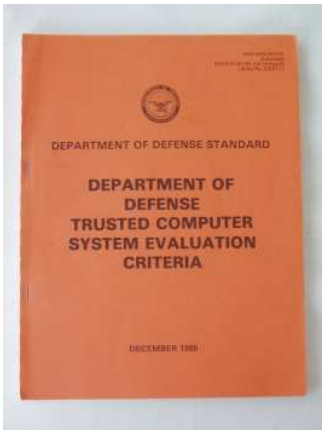

Figura 1.3: Orange Book

A principal característica da DAC é que um recurso possui um ou mais donos e grupos, que especifica quem pode ou não acessar um recurso específico (LAMPSON 1971). Apesar de o *Orange Book* não especificar o conceito de “dono” (*owner*) de um objeto que controla os seus acessos, na prática isso ocorre, provavelmente porque muitos sistemas

que implementam DAC usam esse conceito de “dono”. Nesse modelo, se um usuário é dono de um objeto (um arquivo, por exemplo), ele pode conceder a outros usuários a permissão de acessá-lo em um modo qualquer de operação. Posteriormente, essa permissão pode ser revogada, a qualquer momento.

Com DAC existem apenas duas grandes categorias de usuários: Administradores e Não-administradores. Para alguns serviços e programas rodarem com nível elevado de privilégio, as escolhas são poucas e muitas vezes opta-se pelo completo acesso de administrador. Para os serviços e programas funcionarem de maneira segura, é usado o princípio de dar privilégios mínimos para sua execução. Mesmo assim, essa abordagem não consegue atender as necessidades de segurança de maneira satisfatória, pois ainda assim ela é muito permissiva. Por exemplo, um programa para navegação na internet não precisaria ter acesso a leitura ou escrita arquivos fora do diretório home do usuário, ou até mesmo um servidor web como o apache, não precisaria ter acesso a arquivos sensíveis para o sistema. Isso vai contra o principio de menor privilégio que o DAC tenta aplicar em serviços, expondo a sua fraqueza na aplicação de segurança de programas e serviços. Ainda existem os serviços que devem rodar como super usuário; caso alguma vulnerabilidade nesses serviços seja descoberta e não corrigida, pode comprometer um sistema inteiro.

Atualmente, o DAC é o modelo mais popular de controle de acesso, pela sua utilização em grande escala em sistemas operacionais comerciais. Todas as variantes do UNIX, o Netware e a série Windows NT, 2000, 2003, XP e Vista utilizam o modelo DAC para conceber a implementar as suas checagens de autorização, dispondo também do conceito de grupos de usuários para facilitar na administração e concessão de permissões.

Todos os mecanismos DAC possuem uma falha fundamental que é de não reconhecer a diferença entre usuários humanos e programas de computador, portanto o DAC não está fornecendo ao usuário a habilidade de usar o acesso, ao invés disso, está passando essa habilidade aos softwares. Por exemplo, quando um usuário executa um programa qualquer, o software executado herda todos os poderes de acesso do usuário que o executou. O DAC assume que está em um ambiente em que todos os programas são confiáveis em sem falhas, o que é bem distante do atual mundo real. Além disso, a informação pode ser copiada de um objeto para outro, e a informação pode ser acessada nesse segundo, mesmo que o dono do objeto original não tenha concedido permissão. Com essas fraquezas, o modelo DAC tornou-se insuficiente para sistemas militares, onde os sistemas precisavam ter um alto nível de controle e ser resistente a diversos tipos de ataques. Este foi o principal fator motivador da criação do modelo MAC.

1.1.2 *Mandatory Access Control* - MAC

Mandatory Access Control (MAC), ou Controle de Acesso Mandatório é outro tipo de controle de acesso definido pelo *Orange Book - Trusted Computer System Evaluation Criteria* (NSA 1985) como uma maneira de restringir acesso a um objeto baseado na sensitividade (representada por um rótulo) da informação contida nos objetos.

O MAC é muito mais efetivo que o DAC, pelas seguintes razões:

- MAC não pode ser sobrescrito pelo proprietário do objeto, somente pelo administrador. Enquanto o ponto-chave do DAC é o fato de que os usuários são conside-

rados donos do objeto e portanto responsáveis pelas suas permissões de acesso, o modelo mandatório prevê que usuários individuais não têm escolha em relação a que permissões de acesso eles possuem ou a que objetos podem acessar. Os usuários individuais não são considerados donos dos objetos, e não podem definir suas permissões, isso é realizado pelos administradores do sistema. Assim os usuários não têm o poder de conceder (acidentalmente ou intencionalmente) acesso a dados ou recursos para usuários não autorizados, de acordo com a política definida pelo sistema;

- MAC pode ser aplicado em outros agentes não protegidos pelo DAC além de usuários, como programas, diretórios, *sockets*, tcp *sockets*, unix *stream sockets*, sistemas de arquivos, i-nodos, etc;

- MAC é especificado pelo sistema. Ele não pode ser aplicado, modificado ou removido, exceto por uma operação privilegiada (através do *kernel* ou pelo usuário que obrigatoriamente possuir um papel específico);

- Sobre MAC, cada programa roda sobre uma *sandbox* que limita suas capacidades. *Sandbox* é um mecanismo utilizado para rodar programas de forma segura que provê um conjunto controlado de recursos, como memória, arquivos e rede. O exemplo mais comum de uso de *sanboxes* é o uso de *jails*;

Atualmente existem algumas implementações do mecanimo MAC para o Linux, como: SELinux, AppArmor, GRSecurity, RBAC, e Smack. Esse trabalho irá abordar somente a implementação MAC SELinux (*Security Enhanced Linux*).

2 SELINUX

Mecanismos de controle de acesso MAC têm o objetivo de contornar os pontos fracos de segurança que o modelo DAC possui, porém criar um mecanismo MAC utilizável e flexível tem se mostrado uma tarefa complicada. A principal característica do SELinux é de prover ao Linux um mecanismo flexível, simples para o usuário e configurável de MAC.

2.1 Definição

De acordo com os próprios desenvolvedores do SELinux:

"NSA *Security-enhanced Linux* é um conjunto de *patches* ao kernel Linux que somados a alguns utilitários têm a função de incorporar uma forte e flexível arquitetura de controle de acesso mandatório na maioria dos subsistemas do kernel. Ele provê um mecanismo para forçar a separação da informação baseado em requisitos de confidencialidade e integridade. Ele previne ameaças de adulteração da informação e possibilita o confinamento do estrago que pode ser causado por aplicativos imperfeitos ou maliciosos. Isso inclui o conjunto de políticas exemplo feito para o propósito geral comum da segurança."

Ou de forma mais objetiva, **SELinux** (*Security-Enhanced Linux*) é uma implementação de controle de acesso mandatório (MAC) no *kernel* Linux usando o *framework* LSM (*Linux Security Modules*).

O uso do SELinux tem por objetivos de segurança primários:

- Isolamento das aplicações: busca o nível do menor privilégio no uso de aplicações. Com isso, um problema de segurança em uma aplicação isolada (como *bugs*, vulnerabilidades e *zero-days*) não influencia o sistema como um todo, não comprometendo o funcionamento das outras aplicações. Desta maneira, diminui o efeito da exploração de vulnerabilidades, limitando a propagação de erros e reduzindo a necessidade de aplicação imediata de *patches* de segurança em aplicativos vulneráveis;
- Fluxo de informações: garantia de que a informação deve seguir caminhos predefinidos para acesso entre os processos;
- Confidencialidade: a informação não estará disponível ou será divulgada a indivíduos, entidades ou processos sem a devida autorização;
- Integridade: disponibilidade de informações confiáveis, corretas e dispostas em formato compatível com o de sua utilização. A informação manipulada mantém todas

as características originais estabelecidas pelo proprietário da informação, incluindo controle de mudanças e garantia do seu ciclo de vida (nascimento, manutenção e destruição).

- Auto-proteção: como o SELinux é aplicado diretamente sobre o kernel do Linux, além de proteger as políticas de segurança, ele também tem por objetivo proteger o próprio sistema operacional (binários, configurações, recursos, etc) para se auto proteger;

- Menor privilégio: garante que as políticas aplicadas estão corretas e de que os processos possuem apenas o acesso necessário para realizar a sua função, e nada mais do que isso;

- Separação de papéis: definição das permissões de usuários e processos para evitar a elevação de privilégios e suas consequências.

2.2 Histórico

O SELinux, mesmo se tornando um projeto público a poucos anos, se origina de trabalhos de várias décadas atrás. Em 1973, cientistas da computação David Bell e Leonard LaPadula definiram o conceito de sistema seguro e publicaram um modelo formal descrevendo o sistema seguro multinível (MLS) (BELL 1975).

Nos anos 1980, o trabalho de Bell e LaPadula influenciaram muito o desenvolvedores de sistemas do governo dos Estados Unidos, e acabaram criando o *Orange Book Trusted Computer System Evaluation Criteria* (NSA 1985), definindo 6 classes de avaliação de sistemas seguros: C1, C2, B1, B2, B3 e A1. As classes C1 e C2 englobam sistemas que dependem de controle de acesso discricionário (DAC). Classes B1 e superiores, devem implementar controle de acesso mandatório.

Durante os anos 1990, pesquisadores da *National Security Agency* (NSA) dos Estados Unidos trabalharam junto com a *Secure Computing Corporation* (SCC) e Universidade de Utah para desenvolver uma forte e flexível arquitetura para controle de acesso mandatório. Nessa parceria foi desenvolvido um protótipo da arquitetura, chamada de "*Flask within Fluke*". A arquitetura *Flask* implementa MAC através de uma política de segurança definida que pode controlar todos os objetos do sistema, aplicando o conceito de menor privilégio, que dá a um processo ou objeto exatamente os direitos necessários para executar uma determinada tarefa. O modelo *Flask* habilita a interpretação de uma política de segurança na maneira de um fluxo natural de regras, facilitando o refinamento da política de segurança. Após esse trabalho, a NSA continuou com sua pesquisa, e em dezembro de 2000, publicou junto com a Network Associates, Secure Computing Corporation, Trusted Computer Solutions e Tresys, como um produto de código aberto, a primeira versão do SELinux baseada na arquitetura *Flask*.

Na época da apresentação da solução SELinux pela NSA no encontro de desenvolvedores kernel de 2001, Linus Torvalds, pai do kernel Linux, rejeitou a inclusão do SELinux no kernel de desenvolvimento da época (versão 2.5), por observar que surgiam simultaneamente muitos *frameworks* de segurança diferentes que se baseavam no kernel Linux. Isso levou à conclusão de que não existia um consenso na comunidade de segurança sobre

o modelo a ser adotado e motivou a criação dos LSM (*Linux Security Modules*), que consistem em uma série de "ganchos" (o termo "*hooks*" em inglês) que foram inseridos no kernel oficial do Linux 2.6 em dezembro de 2003 (WRIGHT 2002), enquanto, ao mesmo tempo, o SELinux foi modificado para incorporar LSM.

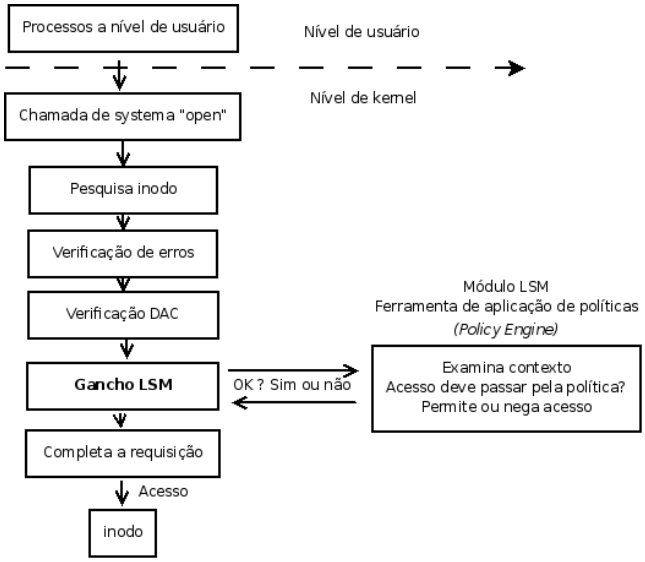

Figura 2.1: Arquitetura de ganchos LSM

A segurança de um sistema Linux não modificado depende da correção do kernel, de todas as aplicações privilegiadas, e de cada uma de suas configurações. Um problema em qualquer uma dessas áreas pode comprometer o sistema por inteiro. Em contraste a isso, a segurança de um sistema Linux com SELinux depende primariamente da correção do kernel e da configuração da política de segurança, e problemas com aplicações individuais ficam limitadas, não comprometendo o sistema por inteiro.

No entanto, implementar um modelo MAC em um sistema unix é muito complexo (MCCARTY 2004). Definir regras para cada usuário para usar cada programa que por sua vez acessa cada objeto resulta num enorme conjunto de regras. Para facilitar, o SELinux utiliza o conceito de RBAC (*Role-Based Access Control*, ou Controle de Acesso Baseado em Papéis). Sobre RBAC, um administrador pode definir papéis e permitir o acesso de alguns usuários a esses papéis. Por exemplo, um usuário pode ter acesso de escrita em seus arquivos pessoais, e um auditor de sistema pode ser capaz de ler todos os *logs* do sistema e arquivos de configuração, mas nunca deverá conseguir escrever neles. A tarefa de definição da política de segurança de controle de acesso mandatório é simplificada, quando:

- se define os papéis em um sistema;
- se especifica quais objetos cada papel pode acessar;
- se associa os usuários a determinados papéis;

De uma perspectiva pura, SELinux provê uma mistura de conceitos e capacidades do controle de acesso mandatório, do controle de integridade mandatório, do controle de acesso baseados em papéis (*Role-based access control* - RBAC), e arquitetura de tipo de

aplicação, que será explicada no decorrer do trabalho.

Existem alguns sistemas operacionais disponíveis com esse nível de controle de acesso, como o Trusted Solaris e o Trusted HP-UX. O objetivo principal do uso do SELinux é demonstrar que esse tipo de sistema operacional de alta segurança é viável além do ambiente governamental e militar, sendo utilizado no campo corporativo e pessoal.

Muitas distribuições Linux começaram a adotar SELinux, mas o primeiro esforço para deixar o SELinux pronto para o uso comercial foi através do projeto Fedora, apoiado pela Red Hat. Hoje diversas distribuições Linux vêm com o SELinux integrado:

- Fedora;
- Red Hat;
- CentOS;
- Engarde.

Existem outras distribuições que, apesar de não vir com o SELinux na sua instalação padrão, estão realizando esforços para permitir a sua aplicação:

- Hardened Gentoo (GENTOO 2007);
- Debian (DEBIAN 2007:1);
- Ubuntu (UBUNTU 2007).

Além do Linux, existem alguns esforços para portar o SELinux para a arquitetura *BSD, como o TrustedBSD (TRUSTEDBSD 2007) e SEDarwin (SEDARWIN 2007).

2.3 Arquitetura do SELinux

A arquitetura do SELinux consiste nos seguintes componentes:

- Código a nível de Kernel: Monitora a atividade do sistema e verifica se as operações requisitadas são autorizadas pela política de segurança corrente, negando qualquer operação não expressamente autorizada. Para realizar esse monitoramento, é utilizada a arquitetura de ganchos LSM, conforme figura 2.2.

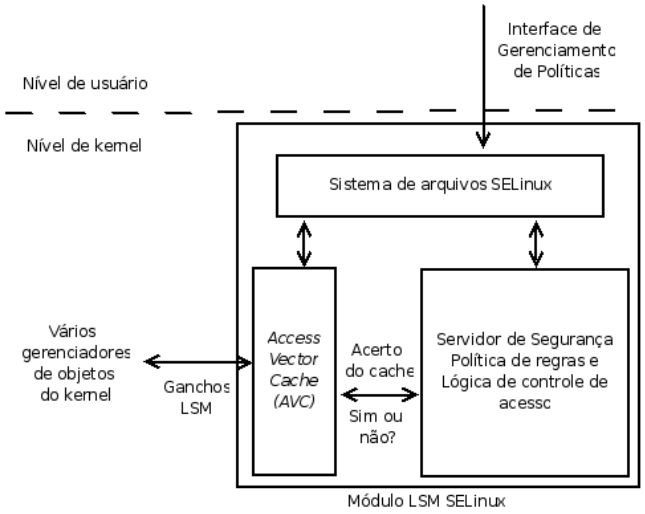

Figura 2.2: Arquitetura do módulo LSM SELinux

O AVC (*Access Vector Cache*) é utilizado para aumentar a performance das verificações das políticas de segurança.

- Biblioteca compartilhada: A maioria dos componentes SELinux são *linkados* com uma biblioteca compartilhada, atualmente chamada libselinux.so.1. Essa biblioteca disponibiliza funções associadas à API do SELinux.

- Política de segurança: O servidor de segurança SELinux baseia suas decisões nos arquivos de políticas que podem ser configurados pelo administrador. A arquitetura do servidor de políticas de segurança pode ser visto na figura 2.3.

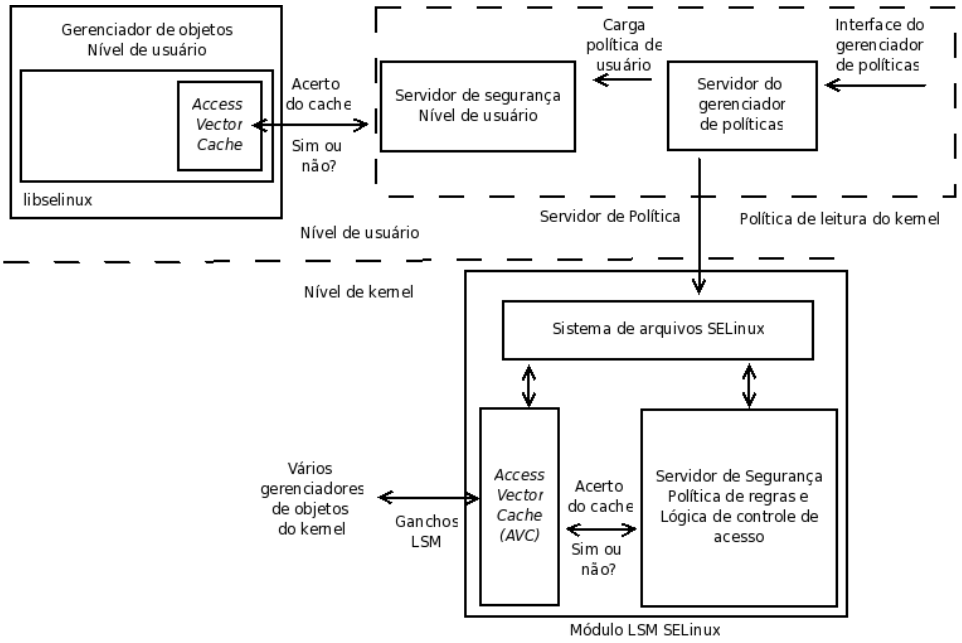

Figura 2.3: Arquitetura do Servidor de Políticas

- Ferramentas: podem ser comandos especiais usados para administrar e usar o SELinux (chcon, checkpolicy, getenforce, newrole, run_init, setenforce, setfiles, avcstat,

getsebool, matchpathcon, selinuxenabled, togglesebool), comandos Linux modificados (ls, id, ps, cp, mv, cron, login, etc) e ferramentas suplementares, com propósito de análise de políticas e desenvolvimento (apol, .setroubleshoot, slide, etc).

- Sistemas de arquivos rotulados: É o sistema de arquivos Linux com suporte a extensão de atributos (xattr).

2.4 Modelo de Segurança SELinux

O modelo de segurança SELinux é baseado em conceitos gramaticais comuns à maioria das linguagens humanas, assim como linguagens de programação de computadores.

2.4.1 Elementos

O modelo de segurança do SELinux é baseado em três elementos: sujeitos, objetos e ações/permissões.

- **Sujeitos**: Sujeitos são os atores em um sistema computacional, mas ao invés do conceito comum de que os usuários são os sujeitos, na verdade, os processos são os verdadeiros atores;

- **Objetos**: Objetos são as potenciais coisas que queremos proteger. Para uma melhor classificação, o SELinux agrupa os objetos em **classes**. O número de classes existentes dependem da versão do SELinux ou do kernel Linux. Novas classes são desenvolvidas para representar novas características do kernel e objetos que podem ser: arquivos, diretórios, sistemas de arquivos, *links*, processos, etc. A relação atualizada de objetos pode ser visualizada em (TRESYS 2007:1).

- **Ações/Permissões**: as ações que sujeitos SELinux realizam em objetos variam de acordo com a classe do objeto. Podem ser: criar, executar, controle de I/O, *link*, pegar atributos, *lock*, ler, renomear, escrever, etc. A relação atualizada de ações/permissões, assim como a relação atualizada de objetos também pode ser visualizada em (TRESYS 2007:1).

Esses elementos são os fundamentos das operações executadas por um servidor de segurança SELinux, e determinam se um sujeito tem permissão de realizar uma determinada ação em um objeto dado. Por exemplo, o SELinux decide questões como: *o processo 4576 possui permissão para ler o arquivo /etc/shadow ?*. Para realizar essas decisões, o servidor de segurança SELinux consulta a base de dados de políticas (*policy database*). A figura 2.4 ilustra esse exemplo.

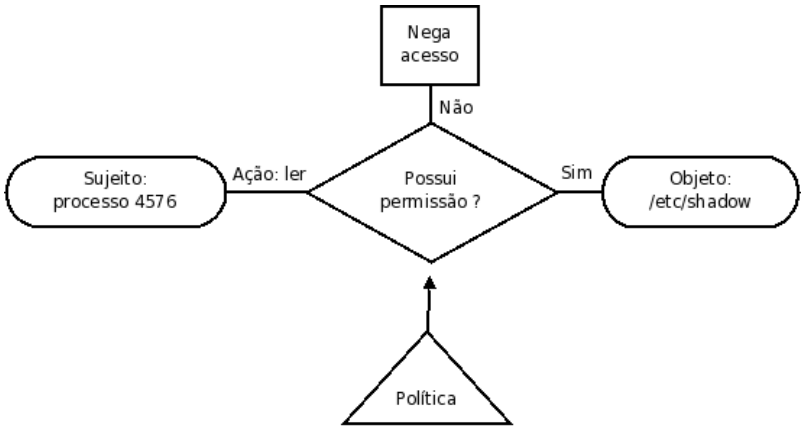

Figura 2.4: Uma decisão SELinux típica

É importante lembrar que a decisão de acesso é sempre feita na seguinte ordem: primeiro a permissão DAC (a permissão comum do Linux) e depois a permissão MAC. Portanto, se o processo do 4576 do exemplo desejar ter acesso ao arquivo /etc/shadow, primeiramente o usuário que rodar esse processo deverá ter a permissão de leitura ao arquivo (DAC) para depois o SELinux tomar a decisão baseando-se nas políticas de segurança pré-estabelecidas (MAC).

2.4.2 Contextos de Segurança

O SELinux aplica a política que é baseada na relação de atributos de um sujeito para atributos de um objeto. Essa relação entre os atributos do sujeito e os atributos dos objetos são referenciados como contextos de segurança. Contextos de segurança são um conjunto de propriedades que são associados à objetos e sujeitos. Essas propriedades são: usuário, papel, tipo e nível/categoria de segurança. A combinação desses campos forma o contexto de segurança. O SELinux armazena os contextos de segurança nos atributos extendidos (xattr) dos sistemas de arquivos, hoje suportado nos sistemas de arquivos mais comuns do mundo Linux: ext2, ext3, reiserfs, xfs, etc. Um Exemplo de um contexto de segurança simples pode ser visualizado na figura 2.5.

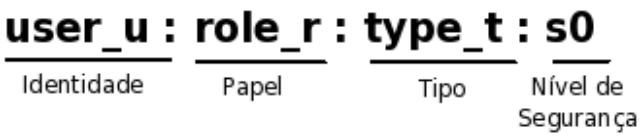

Figura 2.5: Exemplo de contexto de segurança SELinux

A seguir estão detalhados cada um dos campos utilizados na construção do construção do contexto de segurança SELinux:

- **Campo Identidade**: este é o primeiro de quatro campos que forma um contexto de segurança SELinux: O campo identidade ou usuário possui um grupo de usuário SELinux. Se um usuário é membro do grupo “user_u”, então esse campo conterá “user_u”. Processos que são iniciados pelo sistema terão um contexto de segurança iniciando com o campo “system_u”. Grupos SELinux personalizados podem ser criados, por exemplo, se for necessário que um grupo de usuários de domínio tenha acesso a papéis específicos, deve ser criado um grupo de usuários SELinux e definir

quais papéis esse domínio de usuário pode ser capaz de assumir. Em muitos sistemas SELinux, o grupo padrão “user_u” não tem poder de assumir nenhum outro papel, pois o “user_u” é reservado para usuários não privilegiados.

- **Campo Papel**: um papel é um rótulo aplicado a uma identidade. Cada papel contém uma lista de domínios que a identidade têm permissão de realizar transição de rótulos. Rótulos de papéis são convencionalmente nomeados como papel_r (*role_r*). Papéis são usados somente para domínios de usuário, e não para domínios de aplicação, portanto existirá um espaço reservado para informações de papéis nos contextos de segurança para objetos. O campo papel para um processo de domínio de usuário pode variar conforme definido nos grupos SELinux que podem assumir novos papéis. O papel para um domínio de usuário que é membro do grupo SELinux user_u é geralmente user_r. O papel para um domínio de aplicação é system_r. Decisões de segurança baseados nesse campo são referenciados como RBAC (*Role-based access control*).

- **Campo Tipo**: divide sujeitos e objetos em grupos relacionados. Tipos são os atributos de segurança primários que o SELinux usa para realizar decisões de autorização. Qualquer coisa do mesmo tipo terá o mesmo acesso. Os tipos estabelecem *sandboxes* que restrigem processos e previne escalação de privilégios. Por padrão a segurança do SELinux é baseada em tipos aplicados. Isso é chamado de **TE** (***Type Enforcement***). O acesso é determinado pela relação entre o campo tipo do sujeito de origem (um programa binário, por exemplo) com campo tipo do objeto de destino (um objeto do tipo arquivo de senhas, por exemplo). Em SELinux, os termos domínios e tipos são sinônimos. O termo domínio é mais usado para referenciar processos, enquanto tipo é mais usado para referenciar objetos passivos como arquivos.

- **Campo Nível de Segurança**: o SELinux provê suporte opcional à MLS (*Multilevel Security*), que é um tipo de controle de acesso mandatório que é aplicável a alguns problemas de segurança, normalmente associado com controle de dados governamentais classificados, ou seja, dados confidenciais. Esse campo é utilizado para guardar as informações SELinux caso seja aplicado MLS, o que impede que domínios operando em níveis de segurança diferentes possam interagir um com o outro. Caso MLS não seja utilizado, esse campo pode ser descartado, e o valor padrão é assumido (s0), que é nível de segurança padrão que o sistema opera. O nível de segurança s0 também é referenciado como *SystemLow*.

Para verificar o contexto de segurança SELinux de objetos, alguns programas como ls, id, ps, cp, mv, cron, login, etc. foram alterados para incluir a opção “-Z”, para listar os contextos. Como exemplo, para verificar o contexto de segurança de um usuário no prompt, é rodado o seguinte comando:

```
$ id -Z
system_u:system_r:unconfined_t:s0
```

No exemplo, o usuário corrente está no contexto de segurança como usuário system_u, com o papel system_r, no domínio unconfined_t e nível de sensibilidade s0. Ou então, como outro exemplo, para verificar o contexto de segurança SELinux de um arquivo, é usado o seguinte comando:

```
$ ls -Z /usr/bin/passwd
-rwsr-xr-x root root system_u:object_r:passwd_exec_t:s0
/usr/bin/passwd
```

Nesse exemplo, o arquivo /usr/bin/passwd mostra que seu contexto de segurança é do usuário SELinux system_u, com o papel object_r, do tipo passwd_exec_t e nível de sensibilidade s0.

2.4.3 Decisões de Acesso

O servidor SELinux realiza basicamente dois tipos de decisões:

- Decisões de acesso: determinam se um dado sujeito tem permissão de realizar uma dada operação em um determinado objeto (MCCARTY 2004). Esse tipo de decisão é a mais frequente e importante no SELinux. Conceitualmente, cada classe de objeto possui um mapa de bits associado chamando de vetor de acesso, contendo um bit para cada ação definida para a classe, conforme demonstrado na figura 2.6. O servidor de segurança SELinux realiza decisões considerando o contexto de segurança do sujeito, do objeto da ação e a ação requisitada. Quando o servidor de segurança realizar a decisão de acesso, caso ele encontre mais de uma regra da política, ele retornará três vetores de acesso como mostrado na figura 2.6. Na figura 2.6, são demonstradas apenas algumas permissões da classe de objetos arquivo (*file*). Nesse exemplo, o servidor permite ao sujeito adicionar (*append*) ou criar (*create*) o objeto.

	Objeto da classe Arquivo							
	Append	Create	Execute	Get Attribute	Link	Read	Write	...
Allow	X	X						
Auditallow	-	-	-	-	-	-	-	-
Dontaudit	-	-	-	-	-	-	-	-

Figura 2.6: Um vetor de acesso simplificado resultante de uma decisão de acesso

Os três vetores de acesso da figura 2.6 têm as seguintes funções (MCCARTY 2004):

- *Allow*: o vetor de acesso *allow* identifica as operações que o sujeito tem permissão de realizar no objeto. Nenhum registro de *log* é feito nessa operação;
- *Auditallow*: o vetor de acesso *auditallow* provoca um registro no *log* quando a operação é realizada, mas não habilita nenhuma operação, somente o vetor *allow* permite isso.
- *Dontaudit*: o vetor de acesso *dontaudit* identifica as operações que o sujeito não tem permissão de realizar no objeto, mas a negação de acesso não provoca uma entrada no *log*.

Uma ação requisitada é negada a menos que o servidor de segurança retorne *allow*, e requisições que não casam com nenhuma regra da política são negadas, ou seja,

a política padrão do SELinux é **negar tudo**. Se uma ação é negada, é realizada uma entrada no *log* a menos que o servidor de segurança retorne *dontaudit*. Se o servidor de segurança retorna *auditallow*, uma entrada no *log* é gerada, também quando a operação é permitida. Conforme já citado anteriormente, para aumentar a eficiência na realização dessas operações de decisões de acesso, o servidor de segurança SELinux armazena os vetores de acesso frequentes utilizando algoritimos de cache em uma estrutura chamada *Access Vector Cache* (AVC).

- Decisões de transição: determinam os tipos atribuídos a objetos novos criados, particularmente processos e arquivos. Também são chamadas de decisões de rotulagem. Decisões de transição ocorrem em dois contextos comuns (MCCARTY 2004).

  – Criação de processos (sujeitos): o novo processo podem rodar no mesmo domínio que seu pai ou em outro domínio autorizado. Se o processo roda em outro domínio, é realizada uma transição de domínio, conforme exemplificado na figura 2.7 com o *daemon* do serviço bind.

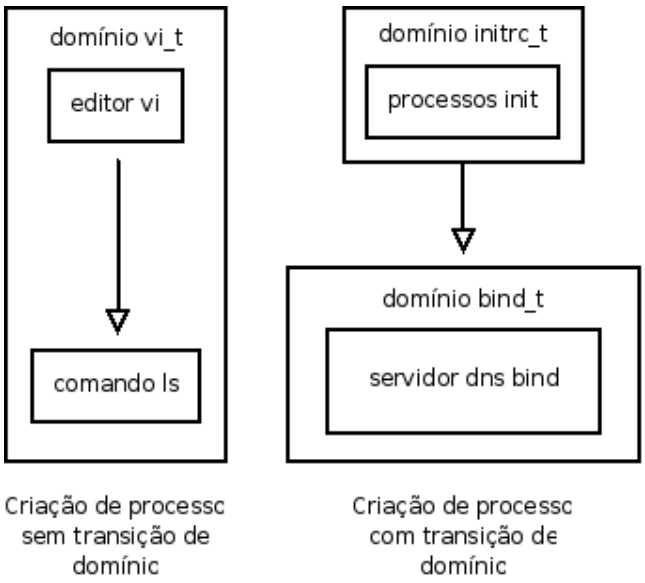

Figura 2.7: Criação de processos e seus contextos de segurança

  – Criação de arquivos (objetos): o novo arquivo (ou um objeto parecido com arquivo, como um diretório) pode ser rotulado com o contexto de segurança do diretório que o contém ou com outro domínio autorizado. Se o contexto de segurança do arquivo possui um outro domínio que não seja o mesmo domínio do diretório que o contém, uma transição do tipo do arquivo é realizada, conforme exemplificado na figura 2.8 com a criação de arquivos pelo serviço syslog no diretório “/tmp”.

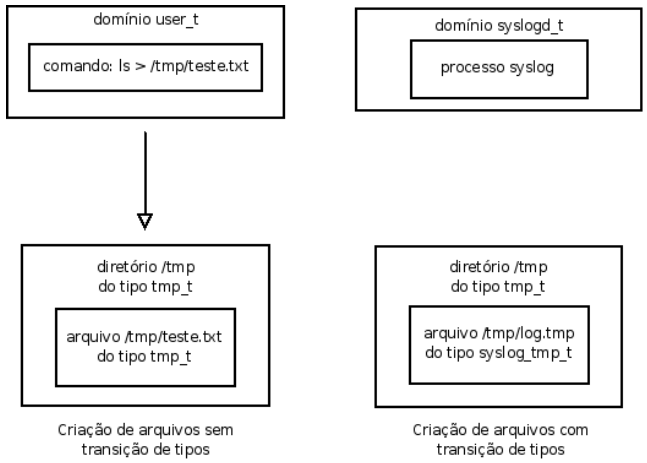

Figura 2.8: Criação de arquivos e seus contextos de segurança

2.5 Instalação

A instalação do SELinux não é necessária caso seja usada uma das seguintes distribuições:

- Fedora Core 2 ou superior: É a base de desenvolvimento do SELinux. Para possuir as últimas versões das políticas e ferramentas SELinux, o Fedora Linux é recomendado, com o preço de que isso pode comprometer a estabilidade, já que é uma distribuição onde essas políticas e ferramentas são testadas antes de serem portadas para o Red Hat Linux. A habilitação do SELinux é opcional durante a instalação, e devem ser selecionadas as opções "*enabled*" e "*enforced*". O suporte é comunitário, através de forums e listas de discussão na internet. As atualizações *on-line* são gratuitas;

- Red Hat Enterprise Linux 4 ou superior: Após os testes no Fedora Linux, as políticas e ferramentas SELinux são portadas para o Red Hat Linux, que além de garantir uma melhor estabilidade, têm a opção de suporte comercial, além do comunitário. A habilitação do SELinux é opcional durante a instalação, e devem ser selecionadas as opções "*enabled* " e "*enforced*". Para conseguir acesso às atualizações *on-line*, é necessária a assinatura paga de um serviço que garante o suporte comercial e este acesso;

- CentOS 4 ou superior: É um clone da distribuição Red Hat Linux, com todas as suas vantagens agregadas, menos o suporte comercial. A habilitação do SELinux é opcional durante a instalação, e devem ser selecionadas as opções "*enabled*" e "*enforced*". Uma vantagem é que as atualizações *on-line* são gratuitas;

- Engarde Secure Linux: É uma distribuição voltada à segurança e possui outras funcionalidades além do SELinux para garantir isso. Possui suporte comercial somente na versão Professional, que é paga. O SELinux já vem pré-habilitado por padrão. As suas atualizações *on-line* são gratuitas;

Como o SELinux está em intenso desenvolvimento, muitas características novas surgiram e outras foram abandonadas, portanto sempre se recomenda usar a última versão

disponível da distribuição Linux. Além disso, é recomendável que se realize as atualizações para que bugs nas ferramentas e políticas de segurança sejam corrigidos.

As seguintes distribuições precisam de interação para que o SELinux seja instalado:

- Hardened Gentoo: Hardened Gentoo não é uma distribuição ou uma solução completa, é apenas um projeto de um grupo de desenvolvedores que têm por objetivo a segurança pró-ativa, através da aplicação de várias ferramentas na distribuição Gentoo Linux. A instalação é feita da mesma maneira que o Gentoo Linux convencional, através de *stages*, porém são utilizados *stages* modificados pelos seus desenvolvedores. O SELinux vêm habilitado por padrão e o suporte é gratuito através da comunidade, via fóruns ou listas de discussão na internet. As suas atualizações *on-line* são gratuitas. Mais informações em (GENTOO 2007);

- Debian: A distribuição Debian Linux não vem com o SELinux em sua distribuição, mas é possível utilizá-lo graças aos esforços de alguns desenvolvedores. O Debian Linux é reconhecido por sua grande estabilidade em servidores, e o suporte é gratuito através de fóruns e listas de discussão na internet. As atualizações *on-line* também são gratuitas e mantidas pela comunidade Debian. Para instruções de como instalar e habilitar o uso do SELinux na distribuição Debian Linux, existem instruções em (DEBIAN 2007:2);

- Ubuntu: A distribuição Ubuntu ainda não suporta oficialmente SELinux, mas alguns testes e tentativas de implementações vêm sendo realizados. O Ubuntu se baseia no Debian, porém não possui a sua estabilidade, mas as versões dos aplicativos são mais atuais. Até o momento, o suporte a SELinux na distribuição Ubuntu está em estágio alpha (experimental), e mais detalhes podem ser vistos aqui: (UBUNTU 2007).

3 ADMINISTRANDO E CRIANDO POLÍTICAS SELINUX

Conforme visto no capítulo anterior, as permissões padrões do Linux (DAC) e SELinux (MAC) são ortogonais. Cada mecanismo usa seus próprios atributos de controle de acesso para realizar uma tarefa permitida, e as duas verificações são feitas primeiramente em relação ao DAC e depois ao MAC. Se um arquivo não tiver a permissão DAC suficiente para o acesso, o acesso será negado imediatamente e a verificação MAC não será realizada.

Política é um conjunto de regras que guia o motor do SELinux. Ela define tipos para objetos e domínios para processos, usa papéis para limitar os domínios, e possui identidades para especificar os papéis que podem ser atingidos por essas identidades. Políticas permitem a flexibilidade de configurar o sistema conforme o propósito, controlando o que cada programa pode fazer, e como programas interagem entre si, considerando o controle de acesso a arquivos, intercomunicação entre processos, etc.

Conforme visto no capítulo anterior, um tipo é uma forma de agrupar ítens baseados em sua semelhança em nível de fundamentos de segurança. Isso não necessariamente tem relação com o propósito de uma aplicação ou o conteúdo de um documento. Por exemplo, um objeto do tipo arquivo pode ter qualquer tipo de conteúdo e ser usado para qualquer propósito ou aberto por seu aplicativo relacionado (como um documento do open office, por exemplo), mas se ele pertence à um usuário e reside em seu diretório *home*, ele pertence a um tipo de segurança específico: **user_home_t**.

Esses tipos de objetos (arquivos na pasta home do usuário) recebem sua equivalência de segurança porque eles são acessíveis na mesma maneira pelo mesmo tipo de sujeitos. Da mesma maneira, processos tendem a ser do mesmo tipo se eles têm as mesmas permissões de outros sujeitos. Na política *targeted*, programas que rodam no domínio unconfined_t possuem um executável com um tipo como sbin_t. Da perspectiva do SELinux, isso significa que eles são equivalentes em termos do que eles podem e do que não podem fazer em um sistema.

Por exemplo, o objeto arquivo binário executável /usr/bin/postgres possui o tipo postgresql_exec_t. Todos os *daemons* alvos possuem seu próprio tipo *_exec_t para seus aplicativos executáveis. De fato, o conjunto inteiro de executáveis do PostgreSQL como createlang, pg_dump, and pg_restore são do mesmo tipo SELinux, postgresql_exec_t, e eles transitam para o mesmo domínio durante a execução, postgresql_t.

A política define várias regras que dizem como cada domínio pode acessar cada tipo.

Somente o que é especificamente permitido pelas regras é autorizado. Por padrão, cada operação é negada e auditada, significando que é gerada uma entrada no log do sistema, como em /var/log/audit/audit.log, no Fedora Linux 8.

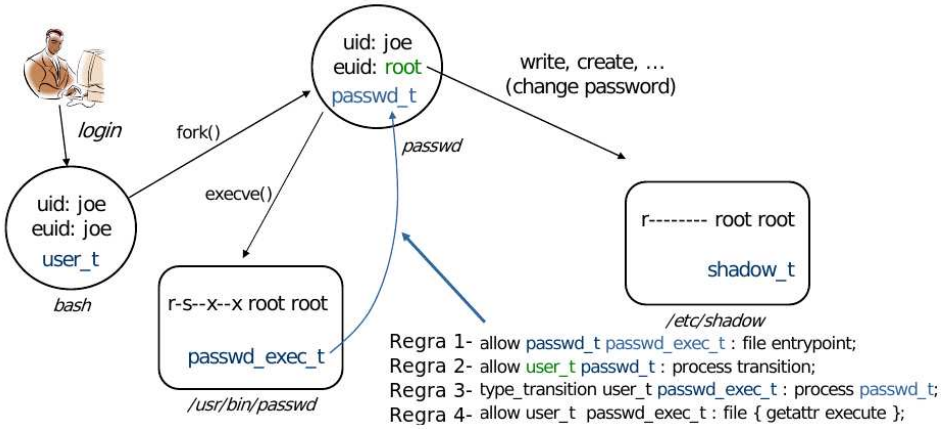

Figura 3.1: Exemplo de política: executável passwd. Fonte: (TRESYS 2007:2)

Na figura 3.1 está demonstrado um exemplo típico do uso de políticas SELinux para permitir transições entre domínios: o usuário joe se loga ao sistema, e tenta trocar a sua senha, que fica armazenada no arquivo /etc/shadow, que é do tipo shadow_t. Ao se logar, o usuário joe possui o contexto de segurança padrão user_t, para trocar a senha, executando o comando /bin/passwd, o processo transita ao domínio passwd_exec_t (permitido na regra 2 na política), e logo após o processo transita ao domínio passwd_t (regra 3 da política). Dessa forma é garantido que o único acesso que um usuário comum pode ter ao arquivo de senhas (/etc/shadow) é através da execução do binário /bin/passwd. No formato padrão DAC do Linux, o binário /bin/passwd possui a permissão set uid (r-s–x–x) que permite que qualquer usuário tenha permissão de super usuário (root, o dono do arquivo /bin/passwd) e possa executar qualquer tarefa com esse binário com essa permissão elevada. Ou seja, basta que o executável /bin/passwd tenha alguma vulnerabilidade conhecida pelo atacante, que ele pode ser explorado para elevação de privilégios de um usuário mal intencionado. Essa é a função das políticas SELinux, limitar o que cada programa pode fazer ou ter acesso.

A política é compilada para um formato binário para ser carregada no kernel do servidor de segurança, e para aumentar a performance, é realizado cache no AVC. Para visualizar as estatísticas de uso do cache do SELinux, é utilizado o comando avcstat:

```
# avcstat
lookups hits misses allocs reclaims frees
68416960 68410650 6310 6310 5264 5811
```

A política pode ser administrativamente definida modificando arquivos existentes no código fonte da política ou adicionando-se arquivos TE (*Type Enforcement*) e arquivos de contexto à árvore da política.

3.1 Parâmetros de kernel SELinux

Existem três parâmetros que podem ser passados em tempo de *boot* (através de interação no menu do boot, ou arquivos de configuração do grub/lilo) que podem mudar a

maneira que o SELinux atua:

- selinux=0: Essa opção desabilita inteiramente o SELinux, o kernel não irá carregar nada da infraestrutura SELinux. Os *scripts* de inicialização irão perceber isso, e criarão o arquivo "/.autorelabel". A criação desse arquivo provoca a rotulação de todos os arquivos na próxima vez que o Linux inicializar com o SELinux habilitado, pois todos os arquivos manipulados e criados com o SELinux desabilitado não recebem rótulos de segurança.

- enforcing=0: Essa opção irá inicializar o sistema em modo permissivo, para fins de resolução de problemas que podem ser provocados pelo uso do SELinux, como políticas e rótulos errados, etc. Nessa opção, o sistema continua rotulando os arquivos corretamente.

- autorelabel=1 Esse parâmetro irá forçar o sistema a realizar a rotulagem de todos os arquivos. Uma opção equivalente a essa seria realizar os comandos: "touch /.autorelabel; reboot". Essa opção é útil quando é realizada a migração da política *strict* para *targeted*, que serão vistas adiante.

3.2 Tipos de Políticas de Segurança SELinux

Basicamente existem três tipos de políticas SELinux: *Strict*, *Targeted* e MLS.

3.2.1 Política *Strict*

Um sistema que usa a política de segurança *strict* é aquele onde tudo é negado por padrão, e é necessário habilitar regras na política de segurança para cada tarefa que cada aplicação ou processo irá realizar. O SELinux foi concebido originalmente para atender à política *strict*, pois as políticas utilizadas são somente de permissão e não de bloqueio. Para um sistema generalista como é o Linux, atender ao uso de uma política *strict* é uma grande e complexa tarefa. As primeiras experiências de uso da política *strict* no sistema operacional Linux foram na distribuição Fedora Core versão 2, onde a política era desabilitada por padrão. Esse novo recurso causou curiosidade em muitos administradores de sistemas, que por desejo de tornar o seu sistema mais seguro, habilitaram o uso da política *strict* durante a instalação. Como a política *strict* ainda não era madura o suficiente para seu uso em sistemas comerciais (o Linux Fedora Core é uma distribuição usada pela Red Hat para testar novos recursos e amadurecimento de versões de aplicações para depois então serem aplicados no Red Hat Linux) e não atendia todos os programas disponíveis na distribuição Fedora, acabaram ocorrendo muitos problemas em seu uso, e esse deve ter sido o fato que motivou a "má fama" atribuída ao SELinux. Ele ficou conhecido por ser um sistema muito complexo, que apenas trazia mais problemas para serem contornados pelos administradores de sistemas. Isso marcou tanto, que a principal pergunta em fóruns de internet sobre segurança e SELinux da época era: "Como desabilito o SELinux?" (WALSH 2005).

Desde então muita coisa mudou no SELinux até os dias atuais, pois hoje a política *strict* se mostra utilizável em sistemas Linux para muitos casos. Os incidentes ocorridos nas distribuições Fedora 2 levaram a criação de uma nova política de segurança SELinux, a *targeted*, e a criação de um novo domínio chamado unconfined_t (WALSH 2006:1).

3.2.2 Política *Targeted*

Através da política *targeted*, cada sujeito e objeto roda em um domínio chamado **unconfined_t** exceto alguns serviços alvos que possuem políticas pré-definidas. Objetos que estão no domínio unconfined_t não possuem restrições e usam a segurança Linux padrão (DAC), ou seja, rodam como se não existisse o SELinux para realizar os controles de segurança. Os serviços que fazem parte da política *targeted* rodam em seus próprios domínios e são restritos à política para cada operação que realizam no sistema, dessa forma se esses serviços forem comprometidos ou de alguma maneira explorados por suas vulnerabilidades, os danos são limitados, podendo até ser controlados.
Como exemplo, o servidor web apache é protegido pela política *targeted*, e não poderá comprometer os outros serviços do sistema, como servidor de nomes, servidor ssh, arquivos privados de usuários, etc, conforme a política pré-definida. No formato padrão DAC, esta proteção não existe. Um bom exemplo disso é a vulnerabilidade no aplicativo gerenciador de conteúdo web chamado Mambo, que na versão 4.0.14 é vulnerável ao ataque descrito na CVE-2005-3738 (CVE 2005). O SELinux é capaz de limitar esse ataque a ponto de torná-lo inútil (MCGUIRE 2007).

A política *targeted* é a padrão na maioria das distribuições Linux que usam SELinux, como Fedora 3 ou superior e Red Hat 4 ou superior. Atualmente existe uma lista muito grande de *daemons* e aplicativos que possuem uma política já definida pelos desenvolvedores do SELinux, não sendo necessária a criação ou modificações dessas políticas (a não ser em casos muito específicos, como uma situação não usual ou até mesmo um comportamento anormal de um aplicativo). Dentre os aplicativos para os quais já existem políticas *targeted* desenvolvidas de acordo com a política de referência (PEBENITO 2006) podem ser destacados: acct, ada, afs, aide, alsa, amanda, amavis, amtu, anaconda, apache, apcupsd, apm, application, apt, arpwatch, asterisk, audioentropy, authbind, authlogin, automount, avahi, awstats, backup, bind, bitlbee, bluetooth, bootloader, brctl, calamaris, canna, ccs, cdrecord, certwatch, cipe, clamav, clock, clockspeed, comsat, consolekit, consoletype, courier, cpucontrol, cron, cups, cvs, cyrus, daemontools, dante, dbskk, dbus, dcc, ddclient, ddcprobe, dhcp, dictd, distcc, djbdns, dmesg, dmidecode, dnsmasq, dovecot, dpkg, ethereal, evolution, exim, fail2ban, fetchmail, finger, firstboot, fstools, ftp, fusermount, games, gatekeeper, getty, gift, gnome, gpg, gpm, hal, hostname, hotplug, howl, i18n_input, imaze, inetd, init, inn, ipsec, iptables, irc, ircd, irqbalance, iscsi, jabber, java, kerberos, kismet, ktalk, kudzu, ldap, libraries, loadkeys, locallogin, lockdev, logging, logrotate, logwatch, lpd, lvm, mailman, mailscanner, miscfiles, modutils, mono, monop, mount, mozilla, mplayer, mrtg, mta, munin, mysql, nagios, nessus, netlabel, netutils, networkmanager, nis, nscd, nsd, ntop, ntp, nx, oav, oddjob, openca, openct, openvpn, pcmcia, pcscd, pegasus, perdition, portage, portmap, portslave, postfix, postgresql, postgrey, ppp, prelink, privoxy, procmail, publicfile, pxe, pyzor, qmail, quota, radius, radvd, raid, razor, rdisc, readahead, remotelogin, resmgr, rhgb, ricci, rlogin, roundup, rpc, rpcbind, rpm, rshd, rssh, rsync, rwho, samba, sasl, screen, selinuxutil, sendmail, setrans, setroubleshoot, slocate, slrnpull, smartmon, snmp, snort, soundserver, spamassassin, speedtouch, squid, ssh, stunnel, su, sudo, sxid, sysnetwork, sysstat, tcpd, telnet, tftp, thunderbird, timidity, tmpreaper, tor, transproxy, tripwire, tvtime, tzdata, ucspitcp, udev, uml, unconfined, updfstab, uptime, usbmodules, userdomain, userhelper, usermanage, usernetctl, uucp, uwimap, vbetool, virt, vmware, vpn, w3c, watchdog, webalizer, wine, xen, xfs, xprint, xserver, yam, zabbix e zebra.

Segundo (WALSH 2007), na versão 8 do Fedora Core Linux lançada em novembro de 2007, a política *strict* deixou de existir como uma política totalmente isolada da política *targeted*. Foi criada uma política híbrida onde tanto a política *targeted* quanto a *strict* é utilizada. No Fedora 8 é possível usar a política *strict* por usuário, adicionando o usuário em um domínio *strict* (domínio *guest* ou*xguest* , por exemplo), e esse usuário estará com limitações de acessos como era antigamente usando-se a política *strict*. Por padrão, os usuários serão criados ainda no domínio unconfined_t. Para remover totalmente a habilidade de processos executarem no domínio unconfined_t, pode-se remover o módulo *unconfined* com o comando "semodule -r unconfined" (WALSH 2007).

3.2.3 Política MLS

Durante o desenvolvimento do Linux Fedora Core 5/Red Hat Enterprise Linux 5, foi desenvolvida a política de segurança MLS, ou *Multilevel Security* (Segurança Multi-nível, em português). A política MLS é similar a política *strict*, mas adiciona um campo nos contextos de segurança para separar níveis. O SELinux pode usar estes níveis para separar dados em um ambiente que necessite de uma separação rígida hierarquicamente. O exemplo mais comum desse tipo de classificação é o no uso militar, onde os dados são classificados em determinados níveis.

Muitas organizações comerciais precisam proteger informações, e a maioria têm algum grau de tolerância para vazamento parcial ou total dessa informação. Organizações que usam sistemas MLS não toleram nenhum tipo de vazamento. O valor da informação que se deseja proteger define o limite de dinheiro que as organizações irão investir em confidencialidade de dados (SMITH 2007). Caso o vazamento da informação ocorra, as principais consequências são perda de negócios e até mesmo o fechamento de empresas. As agências de defesa, que incluem serviços militares, organizações de inteligência, agências relacionadas a governos, não conseguem se recuperar facilmente de alguns tipos de vazamentos de informações. Esses tipos de falhas não podem ser facilmente corrigidos somente gastando mais dinheiro, como no primeiro caso. Durante a Guerra Fria, a ameaça das armas nucleares de aniquilação em massa levou a políticos e militares se preocuparem muito seriamente sobre esse tipo de riscos de vazamento de informações secretas, levando as agências de defesa demandar sobre níveis de segurança de dados muito antes das organizações comerciais precisarem.

O termo multinível é usado porque as agências de defesa classificaram pessoas e informações em diferentes níveis de confiança e sensibilidade. Esses níveis representam as classificações de segurança bem conhecidas: confidencial, secreto, e ultra secreto (*confidential, secret e top secret*). Pessoas que possuem o grau confidencial são autorizados a ver documentos confidenciais, mas não são autorizadas a ver documentos secretos ou ultra secretos. Esses níveis formam uma hierarquia simples demonstrada graficamente na figura 3.2, que ilustra a única direção que os dados devem fluir:

No modelo de segurança MLS, sujeitos e objetos são rotulados com níveis de segurança (SLs - *Security Levels*). Um nível de segurança é composto de dois tipos de entidades:

- Sensibilidade: um atributo hierárquico como "confidencial", "secreto" ou "ultra secreto". Pode assumir valores de s0 até s15 (16 níveis hierárquicos de segurança).
- Categorias: um conjunto de atributos não hierárquicos como "US only" ou "UFO".

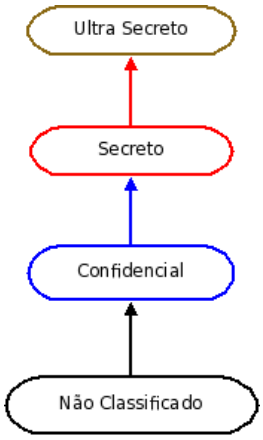

Figura 3.2: Hierarquia dos Níveis de Segurança

Pode assumir valores de c0 até c1023 (1024 categorias de segurança).

Um SL (nível de segurança) possui uma sensibilidade, e pode ter nenhuma ou várias categorias. SLs em objetos são chamados classificações e SLs em sujeitos são chamados autorização. Basicamente o que o modelo de segurança MLS realiza são questões do tipo: “Um processo rodando com a autorização Ultra Secreto / UFO pode escrever em um arquivo classificado como Ultra Secreto / UFO ?” A política implementada do tipo MLS irá responder a esse tipo de questão.

Os nomes das categorias e níveis de sensibilidade são configuradas pelo administrador através do arquivo de configuração /etc/selinux/mls/setrans.conf. Internamente o SELinux enxerga os níveis de sensibilidade apenas como os valores de s0 até s15 e categorias de c0 até c1023. A intenção aqui é habilitar a flexibilidade máxima quando sistemas estão sendo implantados e manter compatibilidade.

Outro conceito importante na política MLS utilizada pelo SELinux é a *VFS Polyinstantiation* (Múltiplas instâncias de Sistemas de Arquivos Virtuais). Muitos programas legados simplesmente não funcionando usando MLS sem *polyinstantiation* porque eles esperam sempre ter autorização para escrever em diretórios como /tmp, $HOME, o que é negado pelo uso de MLS. Para contornar esse problema, são criados múltiplos sistemas de arquivos virtuais com esses diretórios problemáticos para cada programa legado, permitindo assim o seu funcionamento e isolamento total das outras aplicações.

O objetivo da implantação da política MLS no SELinux é permitir o sistema operacional Linux ser certificado EAL4+/LSPP (*Evaluation Assurance Level superior ao nível 4/Labeled Security Protection Profile*). O Linux foi o primeiro sistema operacional que combinou o modelo Bell e LaPadula (BELL 1975) e tipo aplicado (*Type Enforcement*). A certificação LSPP é equivalente à antiga classificação B1 da TC SEC (NSA 2007:1) enquanto a equivalência da certificação EAL 4 ou superior pode ser visualizada de uma forma aproximada a outros métodos de classificação na tabela 3.1 (SMITH 2000):

Tabela 3.1: Equivalências entre níveis de certificação

TC SEC	D	-	C1	C2	B1	B2	B3	A1
IT SEC	E0	-	E1	E2	E3	E4	E5	E6
Common Criteria	-	EAL1	EAL2	EAL3	EAL4	EAL5	EAL6	EAL7

- TC SEC: *Trusted Computer System Evaluation Criteria* é uma norma do Departamento de Defesa dos Estados Unidos (DoD) que estabelece requisitos básicos para avaliar a eficácia dos controles de segurança de um sistema computacional. Também chamado de *Orange Book* (NSA 1985);
- IT SEC: *Information Technology Security Evaluation Criteria* é uma norma criada em 1990 pelos países da França, Holanda, Inglaterra e Alemanha que define os requisitos dos controles de segurança de um sistema;
- *Common Criteria*: *Common Criteria for Information Technology Security Evaluation* é uma norma internacional padronizada (ISO/IEC 15408) para segurança de sistemas. Veio em substituição ao TC SEC (ISO 2005).

Para uma distribuição Linux possuir a certificação EAL4, ela deve ser projetada metodicamente, testada e revisada. Para isso foram desenvolvidos conjuntos de testes, como o LTP (*Linux Test Project*) (LTP 2007) que ajuda a verificar se uma distribuição Linux pode ser submetida à avaliação de um laboratório de testes para efetuar a certificação. Sistemas operacionais comerciais que provêem recursos de segurança baseado em usuário são tipicamente avaliados em EAL4. Exemplos de sistemas avaliados são: Novell NetWare, SUSE Linux Enterprise Server 9, Windows 2000 Service Pack 3 and Red Hat Enterprise Linux 5 (CCEVS 2007).

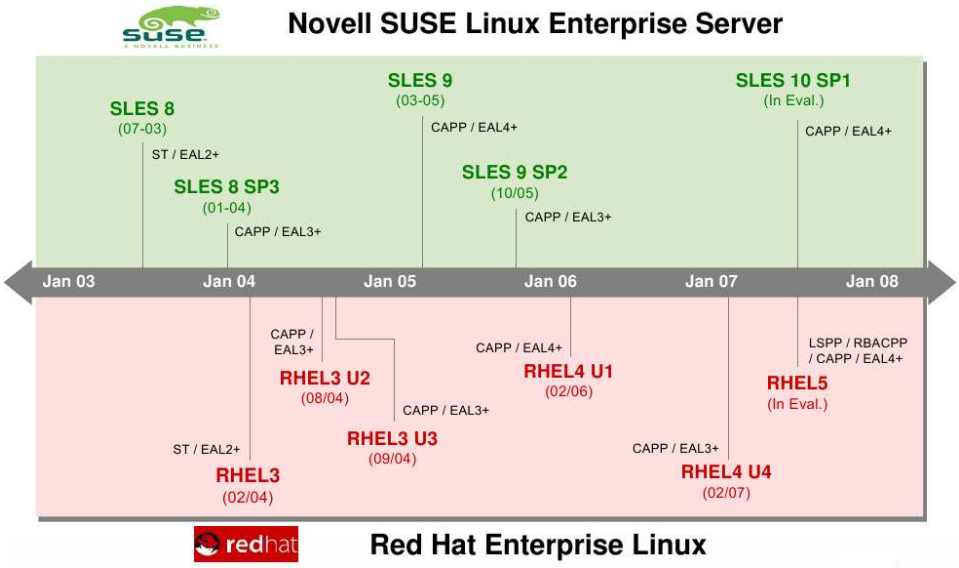

Figura 3.3: Datas das Certificações *Common Criteria* para RHEL e Suse

Uma observação importante é que não necessariamente um sistema utilizando MLS seja mais seguro que outro que não o utilize. O MLS tem como objetivo garantir a confidencialidade e é totalmente dependente da política de segurança para garantir essa característica, enquanto as políticas *strict* e *targeted* têm o objetivo de garantir a integridade.

Por ser uma política de segurança muito complexa e especializada, não é muito utilizada de maneira geral.

Existe ainda um subconjunto do MLS chamado **MCS - *Multi-Category Security*** (Segurança de Múltiplas Categorias, em português). Objetivo da política MCS é adaptar a política MLS para uso mais genérico. As diferenças entre as duas políticas são:

- O nível de sensibilidade não é utilizado, desaparecendo aqui o conceito de múltiplos níveis de segurança (sempre será s0). Geralmente designações hierárquicas de segurança não são muito aplicáveis fora do mundo militar ou ambientes similares, onde são rigidamente definidos e controlados.

- Somente o campo categoria do SL é utilizado (c0..c1023). Categorias podem ser nomes de departamentos, como “Financeiro” e “Recursos Humanos”. Elas devem expressar a natureza da informação e serem associadas com regras de negócio para manuseio de informações.

- O MCS descarta o modelo de segurança Bell La-Padula (BLP). Propriedades BLP como “Não-Escrever-Abaixo”, a qual é designada para previnir vazamento de níveis de segurança altos para níveis de segurança baixos (figura 3.2), pois são na maioria das vezes confusos e podem não permitir o uso de alguns softwares. Por exemplo, considerar que o root não consiga escrever no diretório /tmp é um dos possíveis problemas relacionados a compartilhamento de dados e o uso de MLS.

- MCS é discricionário, similar à lógica DAC do Unix. A implementação corrente permite usuários com controles MCS sobre os seus próprios arquivos. Administradores decidem em quais categorias os usuários estão, e usuários podem adicionar ou remover objetos que possuem, em qualquer categoria que estejam inseridos. É possível configurar o MCS para fazê-lo menos discricionário, mas nesse caso é para isso que o MLS foi feito.

- Para acessar um objeto, as categorias em que o usuário está deverão ser as mesmas, ou um (adequado) superconjunto das categorias do objeto.

- MCS usa a tecnologia MLS, mas não é MLS. MCS não provê nenhuma proteção herdada da política MLS contra softwares maliciosos ou vulneráveis, erros de usuário ou usuários mal intencionados; No entanto DAC e TE são validados primeiro e se o acesso for negado as regras MCS não serão avaliadas.

3.3 Administração de Políticas SELinux

O arquivo de configuração do SELinux é o /etc/selinux/config, e é lido somente em tempo de boot (algumas mudanças necessitam que os arquivos sejam rotulados novamente e sistema seja reinicializado para a configuração nesse arquivo ser efetivada). A biblioteca libselinux lê esse arquivo para descobrir como o sistema está configurado. Dentro desse arquivo, são configuradas quatro varáveis: SELINUX, SELINUXTYPE, SETLOCALDEFS e REQUIRESUSERS.

- SELINUX: Essa variável pode ter um desses três valores: *enforcing*, *permissive* e *disabled*.

  - *enforcing*: Esse deveria ser o modo padrão, diz ao sistema para rodar o SELinux para verificar todos os acessos de sistema e não permitir todos os acessos não autorizados. O kernel bloqueia todos os acessos a menos que o acesso é explicitamente permitido na política. Todos os acessos negados são reportados no sistema de *log* como um AVC (Access Vector Cache), a menos que desenvolvedores da política de segurança tenham criado explicitamente uma regra para não auditar o acesso usando *dontaudit*.
  - *permissive*: O kernel irá reportar todas as violações de acesso na forma de mensagens AVC, mas irá permitir o acesso, e também irá continuar a criar os rótulos dos arquivos de acordo com a política de segurança. O kernel irá gerar os *logs* AVC de uma maneira ligeiramente diferente da usada no modo *enforcing*. Exemplificando: no modo *permissive* apenas o primeiro *log* AVC da violação de acesso de um dado sujeito para um particular objeto é registrado. Já no modo *enforcing*, cada AVC de violação de acesso é registrado no *log*.
  - *disabled*: Diz ao programa de inicialização para desabilitar o SELinux do sistema e pára de criar rótulos apropriados para cada arquivo, ou seja, caso seja necessário reabilitar o uso do SELinux, o sistema precisa recriar os rótulos de todo o sistema (através da criação do arquivo /.relabel e reinicialização da máquina).

  Os valores da variável SELINUX podem ser alterados através de parâmetros passados ao kernel conforme visto anteriormente. Para verificar em qual modo SELinux o sistema corrente está rodando, é utilizado o comando sestatus:

```
# sestatus
SELinux status: enabled
SELinuxfs mount: /selinux
Current mode: enforcing
Mode from config file: enforcing
Policy version: 21
Policy from config file: targeted
```

- SELINUXTYPE: essa variável define em qual diretório verificar os demais arquivos de configuração do SELinux. Esses diretórios são por padrão nomeados de acordo com a política de segurança a ser adotada, portanto pode ser *targeted, strict*, mls, ou mesmo uma política personalizada (minha_política, por exemplo).

- SETLOCALDEFS: diz ao processo de inicialização e à carga da política de segurança se está sendo usada uma nova infra-estrutura de gerenciamento ou não. Se não estiver, ela carrega as customizações locais, variáveis booleanas SELinux e usuários. O valor dessa variável deveria ser sempre 0.

- REQUIRESUSERS: pode ser usado para questionar se existe relação entre usuários SELinux e com alguma entrada padrão. Se não existir, não será possível realizar o *login* e em última análise assume o valor user_u.

É altamente recomendado não alterar os valores das duas últimas variáveis (SETLOCALDEFS e REQUIRESUSERS). Nas distribuições Fedora e Red Hat Linux a configuração do SELinux também pode ser feita através de um utilitário gráfico, o system-config-selinux, demonstrado na figura 3.4:

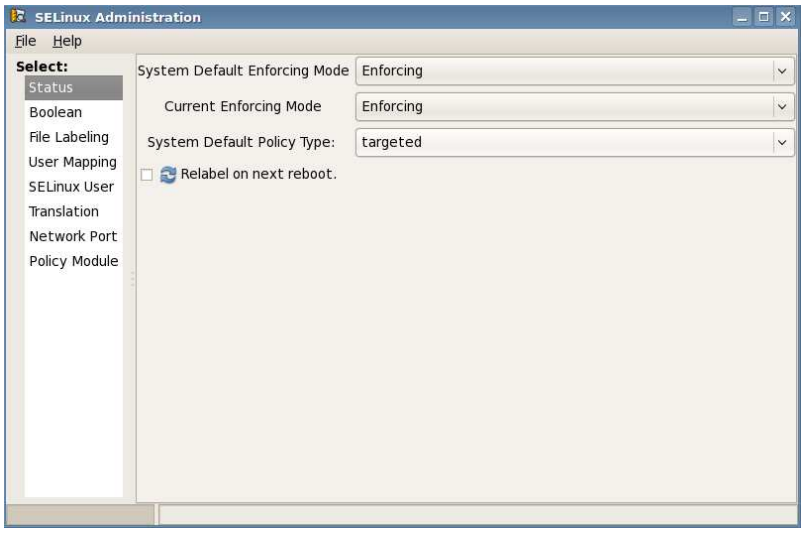

Figura 3.4: Utilitário gráfico para configuração do SELinux

3.4 Carga da Política

A carga da política de segurança é feita durante o processo de inicialização (init), ou pode ser carregada no kernel em tempo real, sem necessidade de reinicialização do sistema. Durante o boot, o SELinux realiza uma tarefa importante: uma vez que todos os processos precisam ser rotulados com o seu domínio apropriado, *init* realiza algumas ações essenciais:

- Depois que o kernel é carregado durante o boot, o processo inicial é associado com o SID (*security identifier*) pré-definido kernel. Identificadores de segurança (SID) são usados para o boot antes que a política é carregada;
- /sbin/init monta /proc, e procura por sistemas de arquivos do tipo selinuxfs. Se ele estiver presente, isso significa que o SELinux está habilitado no kernel;
- Se o init não encontrar o SELinux no kernel, descobre se ele foi desabilitado pelo parâmetro de boot “selinux=0”, ou se o arquivo de configuração do SELinux /etc/selinux/config especifica “SELINUX=disabled”, e o boot continua com um sistema sem SELinux;
- Ao mesmo tempo, init configura o status de enforcing em tempo de execução se ele for diferente do que está em /etc/selinux/config. Isso acontece quando o parâmetro é passado durante o boot. O modo padrão é permissivo até que a política seja carregada, e depois o modo é configurado conforme a configuração no arquivo /etc/selinux/config ou pelos parâmetros de boot enforcing=0 ou enforcing=1;
- Se o SELinux estiver habilitado, o sistema de arquivos /selinux é montado;
- O kernel verifica /selinux/policyvers para versão suportada da política. O processo init verifica em /etc/selinux/config se a política está ativa, como política *targeted* (aplicada), por exemplo, e carrega o arquivo binário de políticas em $SELINUX_POLICY/policy.<version>. Se o binário da política não for a versão suportada pelo kernel, o processo init tenta carregar o arquivo de política da versão anterior para prover compatibilidade com versões antigas da política;

- Se as configurações em /etc/selinux/targeted/booleans forem diferentes daquelas compiladas na política, o processo init modifica a política em memória baseado nessas configurações;
- Os SIDs iniciais são mapeados para contextos de segurança na política, como definido em $SELINUX_SRC/initial_sid_contexts. No caso da política *targeted*, o novo domínio é user_u:system_r:unconfined_t. O kernel pode agora começar a pegar contextos de segurança dinamicamente do servidor de segurança;
- O processo init então se reexecuta a si mesmo para poder realizar transição para um domínio diferente, se a política assim definir. A partir do Fedora Linux 8, para a política *targeted* versão 21, o init muda para o domínio init_t (como constantemente a política está sendo modificada e o SELinux evoluindo, é necessário sempre especificar qual a versão do sistema operacional, distribuição e da política que está sendo usada);
- A partir desse ponto, o processo init continua com o processo normal de inicialização.

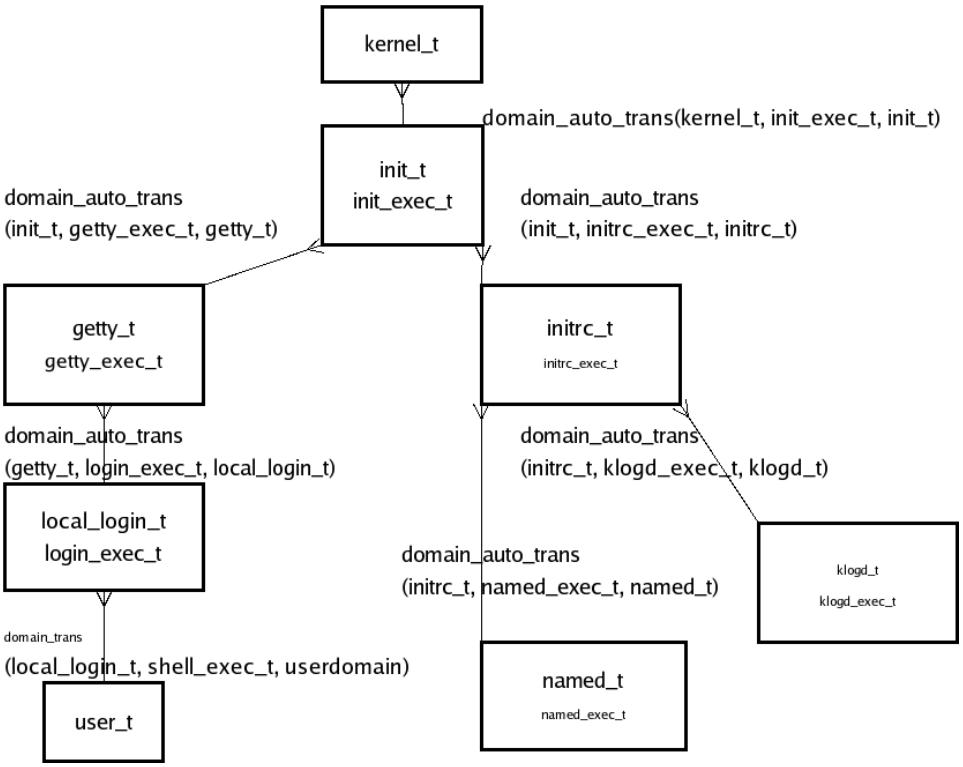

Figura 3.5: Contextos de Segurança *Targeted* na Inicialização do Linux

3.4.1 O Sistema de Arquivos SELinux

O pseudo sistema de arquivos SELinux disponibiliza uma interface primária de controle entre o SELinux LSM em espaço de kernel e os programas em espaço de usuário. Por padrão, o sistema de arquivos SELinux é montado em "/selinux". Muitos utilitários SELinux e APIs (disponibilizadas pela biblioteca libselinux) usam o sistema de arquivos SELinux para acessar o módulo LSM. Muitos arquivos nesse sistema de arquivos são interessantes ao administrador, como:

- Diretório booleans: esse diretório contém um arquivo para cada *boolean* definida na política. O conteúdo do arquivo contém dois valores: o valor corrente e valor pendente. O valor pendente é valor que a *boolean* será modificada quando os valores forem confirmados (*commiteds*, conforme commit_pending_bools). O arquivo possui o mesmo nome que a variável booleana definida na política.

- commit_pending_bools: o arquivo sinaliza ao servidor de segurança em espaço de kernel que a nova política de valores booleanos estão prontos para serem ativados. Essa característica permite que múltiplos valores booleanos sejam modificados em conjunto ao invés de em sequência, caso desejado.

- disable: arquivo de interface que o init utiliza para desabilitar o SELinux durante a inicialização. Somente o init pode utilizar esse arquivo.

- enforce: arquivo de interface utilizado para ligar e desligar o modo aplicado (*enforcing*). Essa é a interface que o init utiliza durante o boot para escolher o modo do SELinux baseado na configuração salva em /etc/selinux/config. Também pode ser utilizado diretamente através dos valores 0 (modo aplicado) e 1 (modo permissivo). A mudança do modo é aplicada imediatamente. O comando setenforce faz exatamente isso de uma maneira mais simples para o usuário.

- load: este arquivo é a interface utilizada pelo programa load_policy para carregar um novo arquivo de política binário.

- mls: usado pelo kernel para indicar se a política MLS está ativada no sistema.

- policyvers: o arquivo contém o valor do número de versão máxima suportado pelo kernel.

3.4.2 Mensagens de Negação AVC

Quando algum processo tenta realizar alguma operação não permitida pela política de segurança vigente e o SELinux está configurado para registrar a tentativa, isso irá gerar uma entrada no arquivo /var/log/audit/audit.log do tipo:

```
Dec 4 17:11:05 rain2 kernel: audit(1165248665.743:5): avc: denied { name_bind}
for pid=11060 comm="vsftpd" src=9281 scontext=system_u:system_r:ftpd_t:s0
tcontext=system_u:object_r:hplip_port_t:s0 tclass=tcp_socket
```

Essa entrada no *log* é chamada de *AVC deny messages*. Alguns ítens no *log* ajudam a identificar o ocorrido, como tipo de acesso negado, número do processo, contexto de segurança de origem, contexto de segurança de destino, classe de objeto e objeto. As mensagens do AVC também podem ser exibidas utilizando o comando: "ausearch -m avc". Apesar dessa entrada do *log* assustar um pouco pela sua complexidade, existe um utilitário que auxilia na sua leitura, como o programa audit2why:

```
# audit2why < /var/log/audit/audit.log
type=AVC msg=audit(1195944021.939:98): avc: denied { noatsecure } for
pid=29135 comm="load_policy" scontext=system_u:system_r:semanage_t:s0 tcon-
text=system_u:system_r:load_policy_t:s0 tclass=process
Was caused by:
Missing or disabled TE allow rule.
Allow rules may exist but be disabled by boolean settings; check boolean settings.
You can see the necessary allow rules by running audit2allow with this audit message
as input
```

A maioria dos erros reportados como AVC são rótulos incorretos nos arquivos. Um exemplo que um label incorreto pode gerar AVCs:

- criação do arquivo resolv.conf no diretório home. Ele irá ficar com o contexto de segurança como esse: "system_u:object_r:unconfined_home_t";
- executar o comando (como root): mv resolv.conf /etc;
- domínios confinados irão reportar com mensagens AVC que estão tentando acessar arquivos do tipo "unconfined_home_t", o que não é permitido pela política;
- para que o arquivo /etc/resolv.conf assuma o seu contexto de segurança correto, é rodado o comando: "restoreconf /etc/resolv.conf". Esse comando irá ler da política de segurança qual é o contexto de segurança padrão configurado para esse arquivo.

Assim como no modelo de segurança DAC, os comandos "cp" e "mv" podem preservar ou não o contexto de segurança e permissão de acesso quando um arquivo é movido ou copiado, conforme a tabela 3.2.

Tabela 3.2: Comportamento dos comandos cp e mv

Comando	**Comportamento**
mv	o arquivo mantém o rótulo original. Isso pode causar problemas como visto no exemplo acima, confusões ou perda de segurança.
cp	cria uma cópia do arquivo usando o comportamento padrão baseado no domínio do processo que o chamou e o tipo do diretório destino. Por exemplo, copiar um arquivo para o diretório *home* de um usuário irá mudar o seu tipo para unconfined_home_t.
cp -p	cria uma cópia do arquivo, preservando os atributos e os contextos de segurança, se possível.
cp -Z <usuário:papel:tipo>	cria uma cópia do arquivo com os rótulos especificados. Sinônimo da opção –context.

Como muitos programas e scripts alteram os arquivos no /etc (como no exemplo o /etc/resolv.conf), foi criado um programa chamado "restorecond". Ele roda como um *daemon* que fica observando se o contexto de algum arquivo que está listado em /etc/selinux/restorecond.conf foi modificado e o restaura automaticamente, tirando esse problema do administrador (WALSH 2006:2).

3.5 Política de Referência SELinux

O projeto de Política de Referência SELinux (*Reference Policy*) é um esforço iniciado pela empresa Tresys para a estruturação da política exemplo da NSA (PEBENITO 2006). O problema dessa política original da NSA é que ela difícil de entender, desenvolver e manter, a menos que o desenvolvedor seja intimamente envolvido com o mecanismo SELinux e conheça profundamente a linguagem de política. Os objetivos para a criação dessa política de referência reestruturada são as seguintes:

- modularidade;
- nivelamento;
- encapsulamento;
- abstração;
- diminuição da complexidade;
- simplicidade de manutenção, modificação e uso;
- garantia de segurança;
- extensabilidade;
- concentração dos esforços;
- prover maior adoção do uso de SELinux.

3.6 Modificando a Política SELinux

A maneira primária de construir um arquivo de política para o kernel é compilá-lo do arquivo de código fonte da política usando o programa **checkpolicy**. O arquivo fonte é tipicamente chamado **policy.conf**

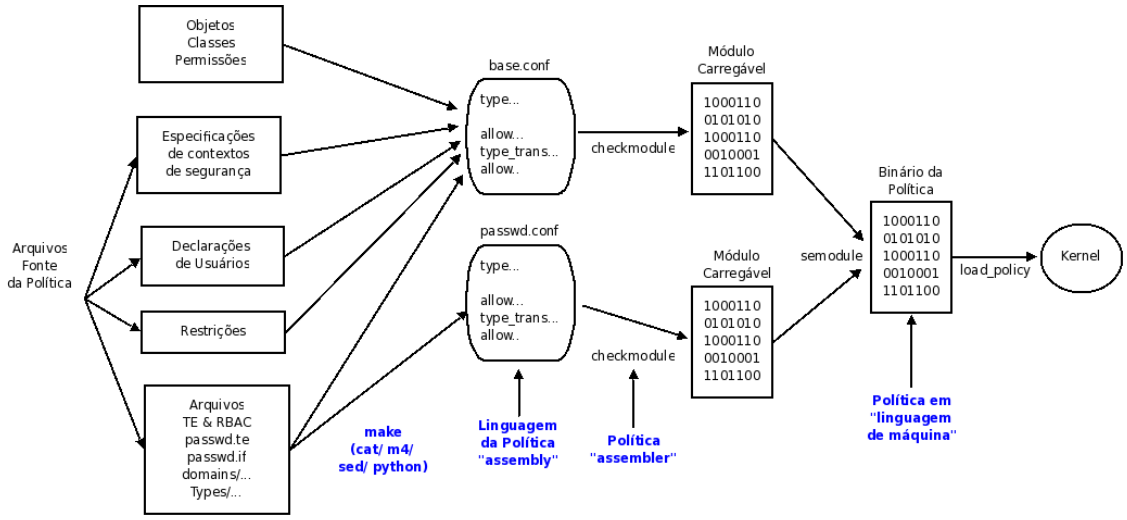

Figura 3.6: Construção de Políticas SELinux

Na verdade, o ideal é que a maioria dos administradores não tenham que modificar ou criar novas políticas SELinux, elas já são pré-definidas pelos desenvolvedores e envolvem

bastante conhecimento sobre fundamentos de segurança, na linguagem da política e até mesmo da aplicação que quer se confinar dentro da política. Apenas em três casos o administrador deverá modificar ou criar políticas SELinux:

- Quando a aplicação que se deseja confinar não é comum, e ainda não existe uma política SELinux definida que possa ser aplicada. Nesse caso deve ser criado um módulo de política especificamente para a aplicação;
- Quando for encontrada alguma característica especial no uso de uma aplicação que não foi prevista pelos desenvolvedores do SELinux, ou alguma personalização que escape do modo usual da aplicação (como rodar em diretórios diferentes dos padrões, por exemplo). Nesse caso a política de referência para a aplicação pode ser modificada para atender à essas personalizações;
- Quando for encontrado algum *bug*, problema de implementação ou interpretação na política SELinux. Nesse caso o *bug* deve ser notificado através da lista "selinux-devel@lists.alioth.debian.org". Antes de notificar o *bug*, verificar sempre se o sistema operacional e as políticas estão atualizados para aa última versão disponível. Como o desenvolvimento e correção das políticas SELinux é constante, pode ser que esse problema já tenha sido resolvido e uma simples atualização corrige o problema sem que o administrador tenha que criar um módulo de política para correção do *bug*.

3.6.1 Variáveis Booleanas

No Linux Fedora Core 3, foi introduzido o conceito de varíaveis booleanas para que o administrador de sistema tenha mais flexibilidade no gerenciamento das políticas SELinux, sem que tenha que criar ou modificar as políticas existentes. São várias declarações do tipo *if-then-else* que foram codificadas diretamente na política que permitem modificar a maneira que o SELinux trabalha somente ligando ou desligando essas variáveis booleanas.

Como as aplicações podem trabalhar de muitas maneiras diferentes, a tarefa do desenvolvedor SELinux de escrever uma política que atenda a todas as situações é difícil. Um bom exemplo de uma situação dessas é o servidor FTP que permite o usuário anônimo (*anonymous*, mas que o administrador deseja que os demais usuários também utilizem o servidor FTP para acessar seus diretórios *home*. Nesse caso torna-se necessária a utilização da variável booleana SELinux ftp_home_dir para habilitar ou desabilitar essa funcionalidade. Para que se tenha uma idéia do poder e da flexibilidade que o SELinux fornece somente com o uso de variáveis booleanas, segue uma relação de algumas variáveis relacionadas somente com o serviço de FTP:

- allow_ftpd_anon_write: permissão de escrita para o usuário anônimo no servidor FTP;
- allow_ftpd_full_access: o servidor FTP pode permitir acesso a todos os sistemas de arquivos visíveis pelo sistema operacional;
- allow_ftpd_use_cifs: permite que o servidor FTP acesse sistemas de arquivos CIFS (samba);
- allow_ftpd_use_nfs: permite que o servidor FTP acesse sistemas de arquivos NFS;

- ftp_home_dir: variável booleana desejada no exemplo, que permite que os diretórios *homes* dos usuários sejam disponibilizados através de FTP.

Além do nome dessas variáveis booleanas que já passam uma idéia da sua funcionalidade, essas variáveis são documentadas em diversos manuais relacionados com os serviços que elas atendem. Por exemplo, para verificar a documentação das variáveis relacionadas ao serviço de ftp, basta dar o comando: “man ftpd_selinux”; as variáveis relacionadas com o serviço http são documentadas através do comando “man httpd_selinux”, etc. Para listar todos os manuais disponíveis, basta executar o comando “man -k selinux”.

Existem duas ferramentas de linha de comando usadas para o gerenciamento das variáveis booleanas:

- setsebool: realiza a mudança de valor de uma variável booleana. Múltiplas variáveis booleanas podem ser modificadas ao mesmo tempo como uma transação, se uma ou mais falhar todas irão falhar. Por padrão, a mudança só terá efeito até o próximo *boot*, a não ser que seja utilizado com a opção “-P ” que torna as mudanças permanentes. Essas mudanças são registradas no arquivo /etc/selinux/targeted/modules/active/booleans.local.

- getsebool: mostra os valores de variáveis booleanas. Com a opção “-a” mostra todas as variáveis. Para verificar todas as variáveis relacionadas com o serviço http, pode ser executado: “getsebool -a | grep http”.

Além dessas ferramentas de linha de comando, existe uma interface GUI onde também é possível realizar mudanças nas variáveis booleanas e ter uma breve explicação do uso de cada uma, através do “system-config-selinux”, mostrado na figura 3.7.

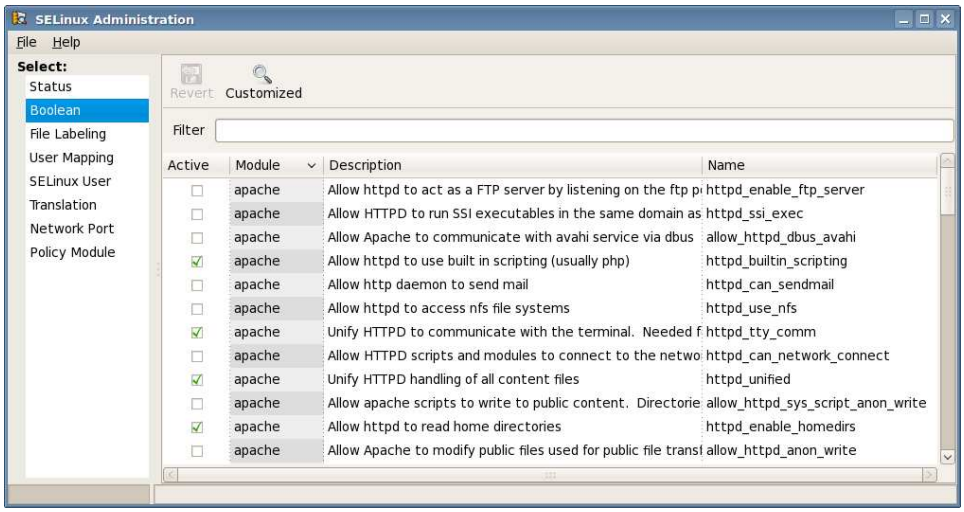

Figura 3.7: Configuração de variáveis booleanas SELinux

3.6.2 Políticas Modulares

Até o Linux Fedora Core 4, o arquivo de política SELinux deveria ser monolítico, ou seja, a política é construída em um único binário pelo checkpolicy e diretamente carregado no kernel. O problema dessa abordagem usando políticas monolíticas é que as políticas SELinux são frequentemente grandes e complexas, e além disso, se a política

fosse alterada era necessário pegar todo código fonte da política, modificá-lo e depois compilá-lo. Em vista disso, a partir do Linux Fedora Core 5 as políticas foram modificas para serem separadas em pequenas unidades chamadas de módulos. Isso significa que desenvolvedores terceiros podem embarcar módulos de políticas SELinux junto com suas aplicações, e eles podem fazer isso sem ter que lidar com todo o código restante da política. Isso atualmente funciona separando-se os passos de compilação e link-edição no procedimento de compilação da política. Módulos de política são compilados dos códigos fonte, e *linkados* quando instalados no "armazém de módulos". Essa política *linkada* é então carregada no kernel para ser aplicada.

O comando primário para lidar com módulos SELinux é o **semodule**, que permite instalar, atualizar e remover módulos. Outros comandos úteis são: **checkmodule**, que é o compilador de módulos incluído no pacote do checkpolicy; e o **semodule_package**, que cria um arquivo de pacote da política (.pp) de um módulo de política compilado. Módulos geralmente são armazenados como um arquivo de pacote da política (extensão .pp) no diretório /usr/share/selinux/$POLITICA/. Lá deve existir pelo menos o arquivo base.pp, que é o módulo básico.

Abaixo seguem os arquivos que compõem o código fonte de um módulo exemplo extraído da Política de Referência SELinux (PEBENITO 2006), aplicado ao serviço BIND (DNS):

- bind.te: contém a política privada completa para o domínio "*named*". A macro "policy_module()" contém o nome e a versão do módulo que serão usados para carregar os módulos de políticas. As próximas nove linhas contém as declarações de tipos e as chamadas de transformações de macro. O tipo para o domínio e seu programa de ponto de entrada são declarados e transformados usando a macro "init_daemon_domain()", a qual é uma interface para o módulo da política do init. O restante da política privada contém as regras para o serviço BIND. Para acessos aos tipos que não estão nesse módulo, existem nove chamadas para interfaces de outros módulos, como por exemplo "logging_send_syslog_msg()" é uma interface do módulo de *log* que permite que o BIND envie mensagens ao serviço de *log*.

```
policy_module(bind,1.1.0)
type named_t;
type named_exec_t;
init_daemon_domain(named_t,named_exec_t)

type named_cache_t;
files_type(named_cache_t)

type named_conf_t;
files_type(named_conf_t)

type named_zone_t;
files_type(named_zone_t)

allow named_t named_cache_t:file manage_file_perms;
allow named_t named_conf_t:file r_file_perms;
allow named_t named_zone_t:file r_file_perms;

kernel_read_system_state(named_t)
kernel_read_network_state(named_t)

corenet_non_ipsec_sendrecv(named_t)
corenet_udp_sendrecv_all_if(named_t)
corenet_udp_sendrecv_all_nodes(named_t)
corenet_udp_sendrecv_all_ports(named_t)
corenet_udp_bind_all_nodes(named_t)
corenet_udp_bind_dns_port(named_t)

logging_send_syslog_msg(named_t)
```

- bind.if: as interfaces públicas do módulo são definidas nesse arquivo. Nesse módulo exemplo, o uso da macro “interface()” define o acesso de entrada para o domínio “named” (como por exemplo, uma transição de domínios). O chamador deverá prover o tipo que ele quer ter acesso para entrar nesse domínio “named”. O arquivo de interfaces também possui uma documentação disponibilizada no formato XML para ser lida por ferramentas de análise ou de desenvolvimento.

```
# <summary>Berkeley internet name domain (DNS) server.</summary>
# <summary>
#
Execute bind in the named domain.
# </summary>
# <param name="domain">
#
Domain allowed access.
# </param>
#
interface('bind_domtrans','
gen_require('
type named_t, named_exec_t;
')

domain_auto_trans($1,named_exec_t,named_t)
allow $1 named_t:fd use;
allow named_t $1:fd use;
allow named_t $1:fifo_file rw_file_perms;
allow named_t $1:process sigchld;
')
```

- bind.fc: contém os contextos dos arquivos. Para facilitar o uso de políticas MLS ou MCS, a macro "gen_context()" foi adicionada. O contexto regular é especificado no primeiro parâmetro, o segundo parâmetro mostra o nível de segurança do arquivo quando uma política MLS é utilizada. O terceiro parâmetro é opcional e especifica as categorias do arquivo quando é utilizada a política MCS, isso possibilita a mudança de políticas MLS para políticas que não sejam MLS, sem modificar a política.

```
/etc/rndc.* – gen_context(system_u:object_r:named_conf_t,s0)
/usr/sbin/named – gen_context(system_u:object_r:named_exec_t,s0)
/var/named(/.*)? gen_context(system_u:object_r:named_zone_t,s0)
/var/named/slaves(/.*)? gen_context(system_u:object_r:named_cache_t,s0)
/var/named/data(/.*)? gen_context(system_u:object_r:named_cache_t,s0)
```

3.6.3 Construindo um Módulo de Política SELinux

Mesmo que não seja uma tarefa comum, o administrador pode se ver obrigado a construir um módulo de política SELinux para atender uma necessidade, seja um erro ainda não corrigido na política atual disponível, até o confinamento na política de um programa específico. Quem se dispor a construir um módulo de política SELinux deve ter conhecimento de como o programa que se deseja confinar funciona e de todos os seus arquivos envolvidos, além de suas implicações no uso de cada arquivo e a comunicação via rede. Supondo que se deseje criar uma política de segurança *targeted* para o programa servidor de SSH (*sshd daemon*), de uma maneira superficial suas funções deveriam ser desmembradas da seguinte maneira:

- permitir que o sshd abra um *socket* tcp na porta ssh (por padrão 22, mas isso é configurável):
  - allow sshd_t ssh_port_t:tcp_socket name_bind;

- permitir que o sshd leia o arquivo de configuração sshd.conf:
  - allow sshd_t etc_t:file read;

- permitir que o sshd leia o arquivo da chave primária de criptografia:
  - allow sshd_t sshd_key_t:file read;

- permitir que o sshd crie o arquivo com o número do processo (PID) na sua inicialização:
  - allow sshd_t var_run_t:dir search add_name ;
  - allow sshd_t sshd_var_run_t:file create write ;
  - type_transition sshd_t var_run_t:file sshd_var_run_t;

Existem ainda muito mais detalhes que devem ser vistos para essa política específica do servidor ssh (o arquivo de interfaces da política atual do servidor ssh possui 416 linhas de regras excluindo-se os comentários e linhas em branco) (PEBENITO 2006).
Visto a complexidade de os módulos de políticas serem criados do zero, e também a dificuldade na interpretação das entradas AVC geradas no *log*, foi criado um software chamado audit2allow. Audit2allow é um *script* feito na linguagem de programação perl que verifica os registros de *logs* AVC negados e gera um conjunto de regras que podem ser adicionadas na política SELinux para liberar essas operações. Ele não tem a intenção de ser um gerador automático de políticas de segurança, mas ajuda muito no desenvolvimento das mesmas. A saída do comando audit2allow deve sempre ser lida com cuidado antes de ser adicionada na política do sistema, porque existe a possibilidade que ele libere mais acessos do que estritamente necessário na aplicação envolvida. Essa ferramenta é melhor utilizada como um guia para escrever políticas.

3.6.4 Customizando Políticas de Segurança SELinux

O conceito de política modular no SELinux permite que desenvolvedores e fornecedores de aplicativos empacotem junto com o próprio aplicativo o módulo de segurança que permitirá a execução dos aplicativos em um ambiente protegido pelo SELinux. A política modular também permite aos administradores do sistema realizar mudanças locais na política de forma mais simples e sem a preocupação com atualizações da política de segurança. A ferramenta utilizada para gerenciar os módulos de política SELinux é o **semodule**. Ele realiza instalação, atualização, listagem e remoção de módulos de política. Também pode ser usado para reconstrução de uma política ou para recarregar a política sem realizar nenhuma outra transação (como na atualização on-line da política de segurança, por exemplo). o semodule atua em pacotes de módulos de políticas criados pelo semodule_package. Convencionalmente, esses arquivos possuem o sufixo ".pp" (*policy package*).

Para verificar os módulos de políticas SELinux instalados, é realizado o comando:

```
# semodule -l
BackupPC 0.0.3
amavis 1.3.1
amtu 1.1.0
apcupsd 1.1.2
audio_entropy 1.1.0
awstats 1.0.0
bitlbee 1.0.0
calamaris 1.2.0
ccs 1.2.0
cdrecord 1.2.1
...
```

Por exemplo de customização da política, vamos supor que o script de inicialização do *daemon* ypbind execute o comando setsebool que tenta usar o terminal e irá gerar a seguinte entrada no *log*:

```
type=AVC msg=audit(1164222416.269:22): avc: denied { use } for pid=1940 comm="setsebool" name="0" dev=devpts ino=2 scontext=system_u:system_r:semanage_t:s0 tcontext=system_u:system_r:init_t:s0 tclass=fd
```

Para eliminar essa entrada AVC, pode ser utilizado o programa audit2allow para gerar uma regra como essa:

```
# grep setsebool /var/log/audit/audit.log | audit2allow
allow semanage_t init_t:fd use;
```

A partir do Fedora 5 e Red Hat Enterprise Linux 5, o programa audit2allow recebeu a possibilidade de construir módulos de políticas que podem ser utilizadas para personalizar a política local utilizada:

```
# grep setsebool /var/log/audit/audit.log | audit2allow -M mysemanage

Compiling policy
checkmodule -M -m -o mysemanage.mod mysemanage.te
semodule_package -o mysemanage.pp -m mysemanage.mod

Para carregar o módulo no kernel:
# semodule -i mysemanage.pp
```

O programa audit2allow irá criar um arquivo do tipo *type enforcement* (mysemanage.te), depois irá executar o comando checkmodule para compilar o arquivo do módulo (mysemanage.mod). Então ele usa o programa semodule_package para criar o pacote da política (mysemanage.pp), e após o comando semodule irá carregar a política no kernel. As mudanças na política serão permanentes e irão sobreviver a reinicializações da máquina. O arquivo pacote da política mysemanage.pp pode ser copiado para outras máquinas e instalado usando semodule.

Para entender como a política foi construída, o arquivo mysemanage.te deverá ser verificado. No exemplo ele terá o seguinte conteúdo:

```
# cat mysemanag.te

module mysemanage 1.0;

require {
class fd use;
type init_t;
type semanage_t;
role system_r;
};

allow semanage_t init_t:fd use;
```

O arquivo mysemanage.te possuirá três sessões. A primeira sessão identifica o nome do módulo, que deverá ser único no sistema, e sua versão. Se for usado o mesmo nome de algum outro módulo já utilizado, o módulo da política atual será substituído, caso o número da versão for maior que o atual. A proxima sessão do arquivo é o bloco "*require*", que é usado para dizer ao carregador da política semodule qual tipo, classes e papéis são requeridos na política do sistema antes que este módulo possa ser instalado. Se qualquer um desses campos estiverem indefinidos, o comando semodule irá falhar. A última sessão é das regras de *allow*. Nesse caso específico, a linha pode ser mudada para "dontaudit", uma vez que o semodule não precisa ter acesso ao *file descriptor* (fd). Por isso é necessário o conhecimento do aplicativo em si e de como as políticas SELinux são construídas, pois apesar do audit2allow facilitar muito a criação de políticas SELinux, ele pode acabar liberando muita coisa sem necessidade, diminuindo a segurança e indo contra ao princípio de menor privilégio.

4 PROTEGENDO SERVIÇOS DE REDE USANDO SELINUX

Para proteger os serviços de rede, será adotado o modo *targeted*, onde apenas os serviços desejados são protegidos pelo SELinux. Devido a complexidade de escrever políticas de segurança SELinux, foi desenvolvida a política de referência (PEBENITO 2006). Essa política de referência tem o objetivo de ser a mais correta e abrangente possível, através de módulos de políticas e variáveis booleanas. Como o Linux é um sistema altamente customizável e pode ser utilizado para vários fins e de várias maneiras, sempre será possível também modificar as políticas de segurança existentes para atender algum problema que possa ocorrer como uma negação de acesso não desejada, ou para uma maior limitação no acesso.

A distribuição usada nos exemplos será o Fedora Core Linux 8, kernel 2.6.23.14-115.fc8 e versão 21 da política SELinux. A configuração do arquivo /etc/selinux/config deverá ficar assim:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
# targeted - Targeted processes are protected,
# mls - Multi Level Security protection.
SELINUXTYPE=targeted
# SETLOCALDEFS= Check local definition changes
SETLOCALDEFS=0
```

4.1 Servidor NTP

O cliente e servidor ntp utilizado já tem sua política definida na política de referência, e não é necessária sua customização. Ao instalar o pacote ntp (realizado via o utilitário yum do Fedora), os arquivos ficaram com os seguintes contextos de segurança:

Tabela 4.1: Contextos de segurança dos arquivos do pacote ntp

Permissão DAC	**Contexto de Segurança**	**Arquivo**
drwxr-xr-x root root	system_u:object_r:etc_t	/etc/ntp
-rw-r–r– root root	system_u:object_r:net_conf_t	/etc/ntp.conf
drwxr-x— root ntp	system_u:object_r:ntpd_key_t	/etc/ntp/crypto
-rw——- root root	system_u:object_r:ntpd_key_t	/etc/ntp/crypto/pw
-rw——- root root	system_u:object_r:ntpd_key_t	/etc/ntp/keys
-rw-r–r– root root	system_u:object_r:net_conf_t	/etc/ntp/step-tickers
-rwxr-xr-x root root	system_u:object_r:ntpd_script_exec_t	/etc/rc.d/init.d/ntpd
-rw-r–r– root root	system_u:object_r:etc_t	/etc/sysconfig/ntpd
-rwxr-xr-x root root	system_u:object_r:bin_t	/usr/bin/ntpstat
-rwxr-xr-x root root	system_u:object_r:bin_t	/usr/sbin/ntp-keygen
-rwxr-xr-x root root	system_u:object_r:bin_t	/usr/sbin/ntp-wait
-rwxr-xr-x root root	system_u:object_r:ntpd_exec_t	/usr/sbin/ntpd
-rwxr-xr-x root root	system_u:object_r:ntpdate_exec_t	/usr/sbin/ntpdate
-rwxr-xr-x root root	system_u:object_r:bin_t	/usr/sbin/ntpdc
-rwxr-xr-x root root	system_u:object_r:bin_t	/usr/sbin/ntpq
-rwxr-xr-x root root	system_u:object_r:bin_t	/usr/sbin/ntptime
-rwxr-xr-x root root	system_u:object_r:bin_t	/usr/sbin/ntptrace
-rwxr-xr-x root root	system_u:object_r:bin_t	/usr/sbin/tickadj
drwxr-xr-x ntp ntp	system_u:object_r:ntp_drift_t	/var/lib/ntp
drwxr-xr-x ntp ntp	system_u:object_r:ntpd_log_t	/var/log/ntpstats

4.2 Servidor FTP

Dependendo do tipo de uso do servidor ftp, algumas personalizações devem ser feitas:

- Caso o ftp tiver capacidade de distribuir arquivos de forma anônima, os arquivos e diretórios compartilhados via ftp devem ser do tipo public_content_t, através da ferramenta chcon:
  chcon -R -t public_content_t /var/ftp.

- Caso seja necessário um diretório onde pode ser feito *upload* de arquivos, esses arquivos e diretórios devem ser do tipo public_content_rw_t:
  chcon -t public_content_rw_t /var/ftp/incoming
  Também é necessário habilitar a escrita de arquivos via ftp anônimo através da variável booleana allow_ftpd_anon_write: setsebool -P allow_ftpd_anon_write=1
  Para que essa configuração seja permanente, por exemplo, sobreviver a um *relabel*, é necessário adicionar uma entrada no arquivo
  /etc/selinux/targeted/contexts/files/file_contexts.local:
  /var/ftp(/.*)? system_u:object_r:public_content_t
  /var/ftp/incoming(/.*)? system_u:object_r:public_content_rw_t

- Por padrão, a política SELinux não habilita que os usuários leiam seus arquivos do seu diretório *home*. Para habilitar isso, a variável booleana ftp_home_dir deve ser configurada:
  setsebool -P ftp_home_dir 1

- O serviço de ftp pode funcionar como um *daemon* ou através do xinetd. Para rodar o servidor ftp como um *daemon*, configurar a variável booleana ftpd_is_daemon: setsebool -P ftpd_is_daemon 1 service vsftpd restart

- Se o servidor ftp publicar arquivos acessados através de NFS ou CIFS, as variáveis booleanas allow_ftp_use_nfs e allow_ftp_use_cifs devem ser habilitadas.

- Se for desejado que todos os arquivos do sistema sejam lidos ou escritos através do servidor ftp, com o controle de permissão verificado somente pelo DAC, a variável booleana allow_ftpd_full_access deve ser configurada com o valor 1.

Para maior documentação sobre o uso de servidores ftp com SELinux habilitado, o comando “man ftpd_selinux” traz mais informações.

4.3 Servidor DNS Bind

Por padrão, a política SELinux não permite que o processo “named” escreva em arquivos de zona mestre, como acontece nos servidores de DNS *slaves*. Caso isso seja necessário (o servidor DNS é *slave*), a variável named_write_master_zones deve ser habilitada:
setsebool -P named_write_master_zones 1

4.4 Servidor SSH

Para o servidor ssh, existem as seguintes variáveis booleanas:

- allow_ssh_keysign: Permite que a autenticação seja feita através de chaves para conexão;

- ssh_sysadm_login: Permite que usuários com rótulo de administradores do SELinux (sysadm_r:sysadm_t) efetuem login via ssh.

4.5 Servidor Web Apache

A política SELinux para o servidor http é muito flexível, possibilitando que os administradores configurem os serviços web na maneira mais segura possível. Os seguintes contextos de tipos de arquivo são definidos para uso com o servidor http:

- httpd_sys_content_t: para conteúdo disponibilizado por todos os scripts httpd e o serviço (*daemon*);

- httpd_sys_script_exec_t: para *scripts* cgi que precisam acessar todos os tipos do sistema (um script em perl, por exemplo);

- httpd_sys_script_ro_t: um arquivo do tipo httpd_sys_script_ro_t só pode ser lido por arquivos do tipo httpd_sys_script_exec_t, bloqueando acessos a dados escritos pelos cgi’s para os outros domínios;

- httpd_sys_script_rw_t: um arquivo do tipo httpd_sys_script_rw_t só pode ser lido e escrito (operações *read/write*) por arquivos do tipo httpd_sys_script_exec_t, bloqueando todos os outros tipos de acesso a dados escritos pelos cgi’s para os outros domínios;

- httpd_sys_script_ra_t: um arquivo do tipo httpd_sys_script_ra_t só pode ser lido e adicionado (operações *read/append*) por arquivos do tipo httpd_sys_script_exec_t, bloqueando todos os outros tipos de acesso a dados escritos pelos cgi's para os outros domínios;

- httpd_unconfined_script_exec_t: permite que *scripts* cgi executem sem a proteção SELinux. Deve ser somente usado em *scripts* httpd muito complexos, depois que todas as outras opções foram exaustivamente testadas e mesmo assim os *scripts* ainda gerarem entradas AVC no *log*;

Também para customização da política do servidor apache, existem as seguintes variáveis booleanas:

- allow_httpd_sys_script_anon_write: usada para permitir que scripts cgi sejam capazes de escrever em arquivos e diretórios compartilhados por outros domínios (Apache, FTP, rsync, Samba);

- allow_httpd_anon_write: permite que o *daemon* do apache escreva em arquivos e diretórios compartilhados por outros domínios (Apache, FTP, rsync, Samba);

- httpd_enable_cgi: permite que o servidor apache execute *scripts* cgi's;

- httpd_enable_homedirs: por padrão, o servidor httpd não tem permissão de acessar os diretórios *home* dos usuários. Caso essa característica seja necessária, é preciso configurar a variável booleana httpd_enable_homedirs e mudar o contexto de segurança dos arquivos que se deseja permitir o acesso para httpd_sys_content_t:
  setsebool -P httpd_enable_homedirs 1
  chcon -R -t httpd_sys_content_t user/public_html

- httpd_tty_comm: por padrão o servidor httpd não tem permissão de acesso ao terminal. Na maioria dos casos isso é desejável, porém pode ocorrer situações onde o servidor httpd precise de acesso ao terminal, como esperar por uma senha para ter acesso a algum arquivo de certificado, por exemplo. Nesse caso, é necessário liberar esse acesso através do comando:
  setsebool -P httpd_tty_comm 1

- httpd_unified: permite que o serviço httpd acesse todos os tipos de arquivos do tipo http (conforme tipos listados acima), ignorando os controles baseados pelo contexto de segurança.

- httpd_builtin_scripting: pode ser usado para desabilitar o uso de *scripting* (PHP). PHP e outros módulos do apache rodam no mesmo contexto de segurança do servidor apache.

- httpd_can_network_connect: por padrão, *scripts* httpd não têm permissão de conectar através da rede. Isso previne que um *cracker* use o servidor httpd para atacar outras máquinas. Se for necessário que os scripts acessem a rede, basta configurar: setsebool -P httpd_can_network_connect 1;

- httpd_can_network_connect_db: permite que *scripts* e módulos tenham acesso a banco de dados pela rede (mysql,postgresql);

- httpd_can_sendmail: habilita ao servidor httpd o envio de e-mails;

- httpd_enable_ftp_server: permite que o servidor httpd aja como um servidor ftp e escute na porta ftp;
- httpd_ssi_exec: permite que *scripts* SSI executem no mesmo contexto de segurança que *scripts* cgi;
- httpd_use_nfs: habilita ao servidor httpd acesso a arquivos e diretórios via nfs;
- httpd_use_cifs: habilita ao servidor httpd acesso a arquivos e diretórios via cifs;
- httpd_can_network_relay: permite que o httpd funcione como um relay (proxy).

5 CONCLUSÃO

Neste trabalho foi demonstrada a necessidade de uma abordagem pró-ativa aos problemas de segurança, visto que os mesmos são inevitáveis de acordo com (LOSCOCCO 1998). A abordagem DAC mostrou-se insuficiente para atender aos requisitos do uso militar e governamental, onde surgiram pesquisas e definições do modelo MAC de segurança e sua implementação em sistemas computacionais. Com a popularização da internet e a disseminação de problemas de segurança, como por exemplo programas com falhas e vulneráveis a ataques, o modelo de segurança MAC se mostrou uma alternativa aos sistemas corporativos e pessoais.

O SELinux foi desenvolvido para diminuir o grau de complexidade da adoção do controle de acesso mandatório, além de trazer a abordagem MAC para sistemas abertos. O SELinux impõe limites às ações dos usuários e programas através de políticas de segurança e tem como base o princípio do mínimo privilégio, onde uma aplicação não terá mais acessos do que aqueles que a permita executar apenas a tarefa a qual ela foi destinada a realizar. Caso essa aplicação possua algum problema de segurança, como uma vulnerabilidade, os estragos que ela pode causar são reduzidos e não comprometem o sistema como um todo. Porém durante o desenvolvimento do SELinux alguns problemas surgiram devido ao fato que o sistema Linux é usado de muitas maneiras diferentes e a política de segurança não era madura o suficiente para o seu uso em aplicações comerciais (WALSH 2005). Desde então muita coisa mudou no SELinux até os dias atuais, pois hoje até mesmo a política *strict* se mostra utilizável para muitos casos. Os problemas iniciais levaram a criação de uma nova política de segurança SELinux, a *targeted*, e a criação de um novo domínio chamado unconfined_t (WALSH 2006:1), que facilitaram muito a adoção do SELinux pelos usuários.

Os demais problemas de complexidade da customização de políticas também foram diminuídos com o uso de variáveis globais e o surgimento de ferramentas que auxiliam o usuário na tomada de decisão quando depara com uma violação da política de segurança, como o audit2why, audit2allow, setroubleshoot, system-config-selinux, etc. Os esforços dos desenvolvedores do SELinux e de suas políticas são constantes e contribuem para que o sistema já se mostre facilmente utilizável e com alto grau de proteção das aplicações. Além disso, surgiram também várias ferramentas que auxiliam os próprios desenvolvedores do SELinux, como: polgen, apol, política de referência, seedit, etc. Essas ferramentas também auxiliam aos desenvolvedores de aplicações que desejam torná-las mais seguras, ajudando na criação e na geração de políticas específicas para uso do SELinux com essas aplicações. E caso isso não seja interesse do desenvolvedor, a aplicação específica sempre poderá rodar no domínio unconfined_t, onde apenas o modelo de segurança DAC é

avaliado, como é mais comum nos dias de hoje.

Levando em consideração todos os aspectos apresentados no desenvolvimento do trabalho, conclui-se que o uso do SELinux para adoção da abordagem de segurança MAC hoje é perfeitamente viável. Esse tipo de sistema que costumava demandar muito tempo de desenvolvimento e até mesmo muito dinheiro, hoje pode ser utilizado por usuários de desktops, passando por sistemas corporativos até sistemas militares e governamentais. A popularização do seu uso está livre dos entraves iniciais que são compreensíveis no início de qualquer implementação de uma nova tecnologia, e a comunidade de desenvolvedores do SELinux se empenhou muito para tornar o SELinux utilizável para todos.

5.1 Trabalhos Futuros

O SELinux mantêm-se em desenvolvimento contínuo, e novas abordagens de sua utilização surgem constantemente, como é comum em projetos de software livre. Hoje a pessoa que se propõe em ajudar no desenvolvimento do SELinux e de novas características dispõe de documentação (a grande maioria em inglês) e ferramentas para auxiliar na formulação de políticas de segurança. Empresas e desenvolvedores individuais envolvidos no projeto do SELinux realizam um simpósio anual onde são propostas novas características e até mesmo linguagens para implementação das políticas de segurança. Os desenvolvedores estão trabalhando para implementar MAC na camada de rede, com integração com IPSEC, iptables, NFS e *clustering*. Além disso, novas ferramentas a nível de usuário, como a setroubleshoot por exemplo, podem ser desenvolvidas para permitir que o usuário tenha um maior controle da política de segurança, emitindo um alerta do tipo: “O programa X tentou obter um acesso indevido ao arquivo Y na pasta Z, isso pode comprometer a sua segurança. Liberar esse acesso? (Sim/Não/Temporariamente)”. Enfim, o SELinux disponibiliza um grande número de possibilidades de desenvolvimento de melhorias e surgimento de novas tecnologias que contribuem para o aumento da segurança de um sistema, e, mais importante, disponibilizando essas tecnologias para usuários comuns.

APÊNDICE A CLASSES DE OBJETOS

Tabela A.1: Classes de Objetos Relacionados à Arquivos

Classe de Objeto	**Descrição**
file	arquivo
blk_file	arquivo de blocos
chr_file	arquivo de caracteres
dir	diretórios
fd	descritores de arquivos
fifo_file	*pipes* nomeados
filesystem	sistema de arquivos formato residente em uma partição de disco
lnk_file	*links* simbólicos ou *hard links*
sock_file	Unix *socket*

Tabela A.2: Classes de Obejtos Relacionados à Comnunicação entre Processos

Classe de Objeto	**Descrição**
ipc	obsoleta, não mais usada
msg	mensagem entre processos em uma fila
msgq	fila de mensagens entre processos
sem	semáforo
shm	segmento de memória compartilhado

Tabela A.3: Classes de Objetos Relacionados à Rede

Classe de Objeto	**Descrição**
association	representa uma associação segura IPSEC
key_socket	IPSEC *socket*
netif	interface de rede
netlink_audit_socket	conexão *netlink socket* de auditoria
ntlink_dnrt_socket	*netlink socket* para controlar roteamento DECnet
netlink_firewall_socket	*netlink socket* para criar *firewalls* em espaço de usuário
netlink_ip6fw_socket	*netlink socket* para criar *firewalls* IPv6 em espaço de usuário
netlink_kobject_uevent_socket	*netlink socket* para enviar notificações de eventos do kernel para o espaço de usuário (por exemplo, temperatura do processador)
netlink_nflog_socket	*netlink socket* para receber mensagens de *log* em espaço de usuário
netlink_route_socket	*netlink socket* para controlar recursos de rede do espaço de usuário
netlink_selinux_socket	*netlink socket* que recebe em espaço de usuário notificações de eventos SELinux
netlink_socket	*netlink socket* que controla todos os *netlink sockets* ainda não especificados em classes SELinux
netlink_tcpdiag_socket	*netlink socket* para monitorar conexões TCP
netlink_xfrm_socket	*netlink socket* para manter parâmetros IPSEC
node	representa um endereço IP ou um intervalo de IPs
packet_socket	pacotes enviados no nível OSI 2
rawip_socket	*sockets* IP que podem ser TCP ou UDP
socket	qualquer tipo de *socket* que ainda não são especificados em classes SELinux
tcp_socket	um *socket* TCP
udp_socket	um *socket* UDP
unix_dgram_socket	datagrama IPC numa máquina local
unix_stream_socket	IPC *stream sockets*

Tabela A.4: Classes de Objetos de Sistema

Classe de Objeto	**Descrição**
capability	privilégios que são implementados como capacidades no Linux, como privilégios do root
passwd	arquivos de senhas Linux (passwd e shadow)
pax	mecanismo de segurança Pax
process	processo
security	servidor de segurança SELinux, existe apenas uma instância dessa classe de objeto
system	o sistema, existe apenas uma instância dessa classe de objeto

APÊNDICE B CONJUNTOS DE PERMISSÕES

Tabela B.1: Permissões comuns de objetos do tipo arquivo

Permissão	**Descrição**
append	adiciona ao conteúdo do objeto
create	cria novo objeto dessa classe
execute	executa o objeto
getattr	acessa os atributos do objeto, como modo de acesso (*stat*)
ioctl	requisita chamadas de sistema no objeto não endereçado por outras permissões
link	cria um *hard link* do objeto
lock	marca de desmarca *locks* no objeto
mounton	usa o objeto como ponto de montagem
quotaon	habilita o arquivo para ser usado como banco de dados de cotas
read	lê o conteúdo do objeto
relabelfrom	muda o contexto de segurança do objeto a partir do tipo existente
relabelto	muda o contexto de segurança do objeto para um novo tipo
rename	renomeia qualquer *hard link* do objeto
setattr	muda os atributos do objeto como modo de acesso (chmod, por exemplo)
swapon	habilita o objeto a ser utilizado como área de troca (obsoleto)
unlink	remove *hard link* do objeto
write	escreve conteúdo no objeto

Tabela B.2: Permissões comuns de objetos do tipo *socket*

Permissão	**Descrição**
accept	aceita uma conexão para o *socket*
append	escreve ou adiciona conteúdo no arquivo de *socket*
bind	nomeia um *socket*
connect	inicia conexão a partir do socket
create	cria um novo arquivo de socket
getattr	consulta os atributos do arquivo de *socket*
getopt	consulta as opções de *socket*
ioctl	controle de I/O de requisições de chamada de sistema no *socket*
listen	espera por conexões ao *socket*
lock	marca de desmarca *locks* no *socket*
name_bind	usa porta ou arquivo
read	lê dados recebidos no *socket*
recv_mesg	permissão requerida para um *socket* receber uma mensagem de uma porta
recvfrom	não usado
relabelfrom	muda o contexto de segurança do *socket* a partir do tipo existente
relabelto	muda o contexto de segurança do *socket* para um novo tipo
send_msg	permissão requerida para o *socket* enviar uma mensagem à uma porta
sendto	envia dados para *sockets* Unix
setattr	muda os atributos de um *socket*
setopt	configura as opções do *socket*
shutdown	interrompe a conexão
write	escreve ou adiciona no *socket*

REFERÊNCIAS

[BELL 1975] D.E. Bell e L.J. LaPadula. **Secure computer systems: Unified exposition and Multics interpretation**. Technical Report ESD-TR-75-306, The Mitre Corporation, Bedford, Air Force Systems Command, Hanscom Field, Beldford, Março 1975.

[BISHOP 2002] Bishop, Matt. **Computer Security: Art and Science**. Editora Addison Wesley; 1136 páginas; Novembro de 2002.

[CVE 2005] Common Vulnerabilities and Exposures List. **Common Vulnerabilities and Exposures (CVE®) List**. Vulnerabilidade CVE-2005-3738, Disponível em <http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-3738>. Acesso em: dezembro 2007.

[CCEVS 2007] The Common Criteria Evaluation and Validation Scheme. **Validated Product - Red Hat Enterprise Linux Version 5 running on IBM Hardware**.Disponível em <http://www.niap-ccevs.org/cc-scheme/st/?vid=10125>. Acesso em: dezembro 2007.

[DEBIAN 2007:1] Coker, Russel; Srivastava, Manoj. **SELinux - Debian Wiki**. Disponível em: <http://wiki.debian.org/SELinux>. Acesso em: outubro 2007.

[DEBIAN 2007:2] Coker, Russel; Srivastava, Manoj. **Debian Wiki - SELinux Setup**. Disponível em: <http://wiki.debian.org/SELinux/Setup>. Acesso em: outubro 2007.

[FERRAIOLO 1992] Ferraiolo, D.F., Kuhn, R. **Role-based access control**. Proceedings of the 15th NIST-NSA National Computer Security Conference, Baltimore, Maryland, USA 13-16 Outubro 1992, p.554–563, Disponível em: <http://csrc.nist.gov/rbac/ferraiolo-kuhn-92.pdf>. Acesso em: outubro 2007.

[FOX 2007] Fox, Tammy. **Red Hat Enterprise 5 Administration Unleashed**. Editora Sams Publishing; 618 páginas; Abril de 2007.

[GASSER 1988] Gasser, Morrie. **Building a Secure Computer System**. Editora Van Nostrand Reinhold; 251 páginas; Maio de 1988.

[GENTOO 2007] Hardened Gentoo Team. **Gentoo Linux Projects - Hardened Gentoo**. Disponível em: <http://www.gentoo.org/proj/en/hardened>. Acesso em: outubro 2007.

[ISO 2005] International Organisation for Standards. **ISO/IEC 15408:2005 Evaluation criteria for IT security**. Suíca, 2005.

[LAMPSON 1971] B. W. Lampson. **Protection**, Proceedings of the 5th Princeton Symposium on Information Science and Systems, pp. 437–443. Princeton University, Princeton, USA, 1971. Reimpresso em Operating Systems Review, p.18–24, Janeiro 1974.

[LEMOS 2006] Lemos, Robert. **SecurityFocus - PHP security under scrutiny**. Disponível em: <http://www.securityfocus.com/news/11430>. Acesso em: dezembro 2007.

[LOSCOCCO 1998] P. Loscocco, S. Smalley, P. Mucklbauer, R. Taylor, S. Turner, J. Farrel. **The Inevilability of Failure The Flawed Assumption of Security Modern Computing Environments**. in Proceedings of the 21st National Information Security Conference, USA, Outubro 1998.

[LTP 2007] Linux Test Project. **Linux Test Project website**. Disponível em: <http://ltp.sourceforge.net>. Acesso em: dezembro 2007.

[LUNT 1988] Teresa Lunt. **Access control policies: Some unanswered questions**. Na IEEE Computer Security Foundations Workshop II, p.227–245, Franconia, Alemanha, Junho 1988.

[GALLAHER 2002] Gallaher, Michael P., O'Connor, Alan C., Krop, Brian. **NIST Planning Report 02-1 - The economic impact of role-based access control**. National Institute of Standards & Technology. Program Office strategic planning and economic analysis group. Março 2002.

[MCGUIRE 2007] Bullington-McGuire, Richard. **Mambo Exploit Blocked by SELinux**. Linux Journal, julho de 2007. Disponível em: <http://www.linuxjournal.com/article/9176>. Acesso em: dezembro 2007.

[MAYER 2006] Mayer, Frank; MacMillan, Karl; Caplan, David. **SELinux by Example: Using Security Enhanced Linux**; Prentice Hall editora; 456 páginas; julho de 2006.

[MCCARTY 2004] McCarty, Bill. **SELinux: NSA's Open Source Security Enhanced Linux**; O'Reilly editora; 254 páginas; outubro de 2004.

[NEGUS 2006] Negus, Christopher. **Fedora 6 and Red Hat Enterprise Linux Bible**. Editora John Wiley & Sons; 1124 páginas; Dezembro de 2006.

[NSA 1985] NSA National Security Agency of USA. **Department of Defense Standard. Orange Book Trusted Computer System Evaluation Criteria**. Dezembro 1985.

[NSA 1999] NSA National Security Agency of USA. **Labeled Security Protection Profile version 1.b**. Disponível em: <http://www.niap-ccevs.org/cc-scheme/pp/PP_OS_LS_V1.B.cfm>. 8 de outubro de 1999; Acesso em: dezembro 2007.

[NSA 2007:1] NSA National Security Agency of USA. **Protection Profile for Multilevel Operating Systems in Environments Requiring Medium Robustness version 1.91**. Disponível em: <http://www.niap-ccevs.org/cc-scheme/pp/PP_OS_ML_MR2.0_V1.91.cfm>. 16 de setembro de 2007; Acesso em: dezembro 2007.

[NSA 2007:2] NSA National Security Agency of USA. **SELinux website**. Disponível em: <http://www.nsa.gov/selinux>. Acesso em: outubro 2007.

[PEBENITO 2006] PeBenito, Christopher J.; Mayer, Frank; MacMillan; Karl. **Reference Policy for Security Enhanced Linux**. Proceedings of the Second Annual Security Enhanced Linux Symposium, Baltimore, Maryland, USA, 28 de Fevereiro a 3 de Março 2006; Disponível em: <http://selinux-symposium.org/2006/papers/05-refpol.pdf>. Acesso em: dezembro 2007.

[SANDHU 1996] Sandhu, Ravi S. **Access Control: the Neglected Frontier**. Proceedings of First Australian Conference on Information Security and Privacy, 1996.

[SHIFLETT 2005] Shiflett, Chris. **Essential PHP Security**. Editora O'Reilly Media; 124 páginas; Outubro de 2005.

[RAVI 1996] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein e Charles E. Youman. **Role-based access control models**. IEEE Computer, p.38–47, Fevereiro 1996.

[REDHAT 2006] Red Hat Inc. **Red Hat Enterprise Linux Deployment Guide**. Disponível em: <http://web.mit.edu/rhel-doc/5/RHEL-5-manual/Deployment_Guide-en-US/index.html>. Acesso em: dezembro 2007.

[SEDARWIN 2007] Sparta Inc. **SEDarwin: Security Enhanced Darwin**. Disponível em: <http://www.sedarwin.org>. Acesso em: outubro 2007.

[SMITH 2000] Smith, Richard. **Trends in Government Endorsed Security Product Evaluations**. White paper Disponível em: <http://www.smat.us/crypto/docs/evaltrends.pdf>. Acesso em: dezembro 2007.

[SMITH 2007] Smith, Richard. **Introduction to Multilevel Security**. Disponível em: <http://www.cs.stthomas.edu/faculty/resmith/r/mls/index.html>. Acesso em: dezembro 2007.

[TRESYS 2007:1] Tresys Tecnology LLC. **Tresys website**. Disponível em: <http://www.tresys.com/selinux/obj_perms_help.html>. Acesso em: novembro 2007.

[TRESYS 2007:2] Tresys Tecnology LLC. **Security Policy Development Primer Course for Security Enhanced Linux**. Disponível em: <www.tresys.com/files/course>. Acesso em: novembro 2007.

[TRUSTEDBSD 2007] TrustedBSD Developer Team. **TrustedBSD Project**. Disponível em: <http://www.trustedbsd.org>. Acesso em: outubro 2007.

[UBUNTU 2007] Sellers, Chad; Case, Caleb. **HardySELinux**. Disponível em: <https://wiki.ubuntu.com/HardySELinux>. Acesso em: novembro 2007.

[WALSH 2005] Walsh, Daniel J. **Targeted vs Strict policy - History and Strategy**. Proceedings of the 2005 Security Enhanced Linux Symposium, Baltimore, Mariland, USA . Disponível em: <http://selinux-symposium.org/2005/presentations/session4/4-1-walsh.pdf>. Acesso em: outubro 2007.

[WALSH 2006:1] Walsh, Daniel J. **Dan Walsh Live Journal - Unconfined Domain**. Disponível em: <http://danwalsh.livejournal.com/9031.html>. Acesso em: outubro 2007.

[WALSH 2006:2] Walsh, Daniel J. **Dan Walsh Live Journal - Introducing restorecond**. Disponível em: <http://danwalsh.livejournal.com/4368.html>. Acesso em: dezembro 2007.

[WALSH 2006:3] Walsh, Daniel J. **Dan Walsh Live Journal - How do I make local customizations in RHEL 5?** Disponível em: <http://danwalsh.livejournal.com/8637.html>. Acesso em: dezembro 2007.

[WALSH 2007] Walsh, Daniel J. **Fedora Developer Interview - Daniel Walsh**. Disponível em: <http://fedoraproject.org/wiki/Interviews/SELinux>. Acesso em: dezembro 2007.

[WRIGHT 2002] Wright, Chris; Cowan, Crispin; Morris, James; Smalley, Stephen; Kroah-Hartman, Greg. **Linux Security Modules: General Security Support for the Linux Kernel**. Proceedings of the 11th USENIX Security Simposium, Baltimore, São Francisco, Califórnia, USA 5-9 Agosto 2002. Disponível em: <http://www.usenix.org/event/sec02/full_papers/wright/wright.pdf>. Acesso em: outubro 2007.