Paradigmas de Segurança em Sistemas Operacionais

Diogo Ditzel Kropiwiec[1]*, Paulo Lício de Geus[1]

[1]Laboratório de Administração e Segurança de Sistemas
Instituto de Computação – Universidade Estadual de Campinas
Caixa Postal 6176 – 13084-971 Campinas, SP - BRASIL

{diogo,paulo}@las.ic.unicamp.br

Abstract. *Security has become one of the principal concerns during the development of applications in general. However, the continual growth in the number or security incidents reported demonstrates that such efforts have not been sufficient to contain the hacker's advances. In this paper, the security paradigms used by the more common operating systems are presented and their vulnerabilities discussed, highlighting the factors that have contributed to the growing number of attacks. New security paradigms are also presented and the factors that hinder their fast adoption are analyzed.*

Resumo. *A segurança tem se tornado um dos principais focos no desenvolvimento de aplicações em geral. O crescimento do número de incidentes de segurança, entretanto, demonstra que os esforços estão sendo insuficientes para conter o avanço dos hackers. Neste artigo, são apresentados os paradigmas de segurança sobre os quais se baseiam os sistemas operacionais de uso mais comum, e suas falhas, alertando assim para os motivos que tem levado ao crescimento do número de ataques. São apresentados, também, novos paradigmas de segurança analisando os aspectos que impedem uma rápida adoção dos mesmos.*

1. Introdução

Os sistemas operacionais surgiram com dois objetivos principais: criar uma camada de abstração entre o *hardware* e as aplicações, e gerenciar os recursos de forma eficiente [Tanenbaum and Woodhull, 1997]. Na primeira etapa de desenvolvimento, as preocupações de segurança com relação a esses sistemas operacionais eram mínimos, se não inexistentes, visto que os computadores estavam isolados, e a única forma de invasão possível era a presencial, isto é, fazia-se necessário estar em frente ao terminal para perpetrar alguma atividade maliciosa. Adicionalmente, o poder de processamento dos computadores era muito reduzido, e o tempo a ser gasto com outras atividades que não os objetivos primários—abstração de *hardware* e gerenciamento de recursos—era altamente custoso.

Com o passar do tempo, os computadores, assim como os sistemas operacionais, foram se aprimorando e se disseminando. Organizações com vários computadores passaram a montar redes para interligá-los, e os sistemas operacionais passaram a suportar a execução de múltiplas tarefas simultaneamente, permitindo que duas

*Apoio CNPq.

ou mais pessoas compartilhassem o tempo de processamento de uma mesma máquina [Garfinkel and Spafford, 1996, Tanenbaum and Woodhull, 1997]. Em paralelo, a capacidade de processamento cresceu enormemente, e aplicações mais elaboradas e complexas puderam ser desenvolvidas.

Com o advento da Internet, muitas dessas redes organizacionais se uniram. A troca de informação foi amplamente beneficiada, e as transações comerciais entre organizações foram simplificadas e ampliadas. Usuários domésticos passaram a fazer parte das redes, e uma vasta quantidade de informações tornou-se acessível a qualquer pessoa [Nakamura and de Geus, 2002].

Infelizmente, essa expansão computacional e a facilidade de se obter informações trouxe um problema esquecido nos primeiros momentos: a segurança das informações. Dados recentes mostram claramente que o número de incidentes de segurança reportados aumenta a cada ano, de apenas 2.412 em 1995, para 137.529 em 2003 [CERT, 2003]. Entretanto, a preocupação com segurança vem acompanhando o desenvolvimento dos sistemas operacionais e suas aplicações há muito tempo, como pode ser visto em [Harrison et al., 1976], contrariando o que as estatísticas parecem demonstrar. Quase 30 anos se passaram, e mesmo todos os esforços voltados a eliminar as vulnerabilidades têm se mostrado insuficientes.

Como primeiro passo de um trabalho maior em desenvolvimento, este artigo analisará os paradigmas de segurança de computadores, tanto os presentes nos sistemas operacionais de uso corrente quanto os presentes em sistemas operacionais ainda restritos ao meio acadêmico, bem como suas limitações e complicações. O objetivo é entender o que tem causado a proliferação contínua dos problemas de segurança noticiados atualmente e o que pode ser feito para reverter esse quadro.

Na Seção 2 serão abordados os paradigmas de segurança presentes nos sistemas operacionais de uso corrente. Na Seção 3 serão apresentadas as falhas desses paradigmas, que levam ao quadro apresentado acima. Na Seção 4 serão apresentados os novos paradigmas de segurança, dos quais já existem implementações disponíveis para uso, mas que como será visto na Seção 5 ainda devem demorar a tornarem-se padrão. Finalmente, na Seção 6, serão apontados os próximos focos de pesquisa necessários para viabilizar a adoção dos novos paradigmas, visando robustecer a segurança dos sistemas operacionais.

2. Paradigmas de Segurança dos Sistemas Operacionais de Uso Corrente

Por sistema operacional de uso corrente, estão sendo englobados os sistemas operacionais mais difundidos em organizações e computadores pessoais. Neste grupo, são enquadrados os derivados do Unix (Linux, BSD e suas várias distribuições), o Windows (com destaque ao NT, 2000 e XP) e o MacOS. Esses sistemas são derivados dos sistemas operacionais de terceira geração [Tanenbaum and Woodhull, 1997], em especial o Unix—no qual as preocupações de segurança se restringiam aos poucos usuários que tinham acesso ao computador, diretamente via terminal, ou pela rede interna isolada à qual pertencia a máquina—e o DOS—no qual preocupações de segurança de recursos não existiam.

A segurança desses sistemas operacionais reside nos direitos de acesso de um usuário (ou grupo de usuários) a um recurso (arquivo, periférico, memória, etc.), baseado

no conceito de permissões de acesso em anéis do Multics [Feiertag and Organick, 1971], e posteriormente adaptado a usuários no Unix [Ritchie and Thompson, 1974]. Neste último artigo, o próprio autor alerta que o modelo de segurança proposto é simples, porém adequado às suas necessidades. O Windows apresentava originalmente uma segurança bem inferior a essa, por utilizar como sistema de suporte o DOS (e conseqüentemente, o sistema de arquivos FAT16/32, que não provia nenhum tipo de controle de acesso), e só posteriormente o Windows veio a utilizar um sistema de arquivos que permitia o controle de acesso dos usuários, o NTFS.

Serão apresentados agora três paradigmas comumente presentes nos sistemas operacionais, que serão numerados apenas para simplificar as explicações subseqüentes sobre os mesmos.

O primeiro paradigma de segurança presente nesses sistemas consiste do usuário como limitador de segurança. O usuário, ao criar arquivos e aplicações, associa a estes permissões referentes a leitura, escrita e execução—em alguns modelos, essas permissões são expandidas a um conjunto maior com ações mais especificas—para si próprio, para usuários do mesmo grupo, e para os demais usuários do sistema [Garfinkel and Spafford, 1996], e este conjunto de permissões associados aos arquivos será denominado domínio. Dessa forma, fica sob a responsabilidade do usuário restringir os arquivos/recursos que julgar relevantes. De forma similar, ao executar uma aplicação, esta terá acesso ao domínio do usuário, podendo interagir com aplicações e/ou arquivos da mesma forma que o usuário do qual obteve o domínio. Adicionalmente, existem situações em que se deseja permitir que um usuário, por intermédio de uma aplicação apropriada, possa interagir no sistema operacional como se fosse outro usuário, e nestes casos são associadas *flags* [1] que permitam o chaveamento de direitos [Garfinkel and Spafford, 1996].

O segundo paradigma de segurança consiste na concentração e isolamento dos direitos privilegiados do computador. Os sistemas operacionais possuem um usuário, denominado super-usuário ou administrador, que tem acesso a todos os recursos da máquinas, e ao qual são restritos todas as operações privilegiadas [Garfinkel and Spafford, 1996]. É importante ressaltar que o domínio do super-usuário compreende acesso irrestrito a todos os recursos do computador, inclusive os arquivos e aplicações pertencentes aos demais usuários. Aos usuários comuns e suas respectivas aplicações só é permitido o acesso a um subconjunto dessas operações privilegiadas, e ainda assim limitado pelas decisões do núcleo do sistema operacional, através de uma interface bem definida com o sistema operacional, conhecida como *API - Application Programming Interface*. Dessa forma, garante-se que os usuários não possam causar alguma falha de operação do sistema operacional, seja acidentalmente, seja intencionalmente. Como pode ser observado, o acesso às operações privilegiadas é total ou nenhum, já que a *API* é responsável por controlar o acesso indireto realizado pelas aplicações. Nesse contexto, existem aplicações, que por necessitarem de acesso a alguma operação privilegiada não provida pela *API*, acabam tendo que usar o domínio do super-usuário, tendo acesso irrestrito a todas as operações privilegiadas.

Um terceiro paradigma comumente usado consiste na segurança pela cifragem

[1] No Linux, por exemplo, existe o *bit* SetUId associado às aplicações. Quando ativo, a aplicação é executada no domínio do seu proprietário, e não do usuário que comandou a execução

das informações. Existe um variado conjunto de técnicas de cifragem, empregadas tanto para mensagens a serem enviadas pela rede de forma segura [Tanenbaum, 2003] quanto para armazenar os arquivos em disco rígido [Silbertschatz et al., 2002]. Com a cifragem, o usuário tem a garantia de que, mesmo que alguém consiga acesso não-autorizado ao arquivo ou mensagem, esta será ilegível sem a chave de criptografia apropriada. A única vulnerabilidade a cifragem é o super-usuário que, durante o processo de cifragem, pode acessar a memória irrestritamente e, conseqüentemente, obter a mensagem/arquivo, ou ainda a chave de criptografia. Encerrada a aplicação de cifragem, mesmo o super-usuário não terá acesso ao conteúdo, caso não tenha armazenado a chave.

Derivado desses paradigmas surgiu o conceito de que as aplicações devem prover a segurança que está faltando e que podem adequadamente garantir tal segurança sem auxílio do sistema operacional [Loscocco et al., 1998]. Uma vez comprometida uma aplicação, todo o domínio ao qual estava associada a aplicação estará comprometido. Por esse motivo, toda vez que uma vulnerabilidade é descoberta, são disponibilizadas correções (denominados comumente de *patches*) à aplicação comprometida.

3. Falhas do Paradigmas Atuais

Os paradigmas apresentados são eficientes se restritos ao contexto em que foram desenvolvidos, com computadores isolados ou pequenas redes isoladas. Nestes casos, o número de atacantes potenciais é extremamente reduzido, restrito às pessoas que têm acesso às máquinas. Entretanto, o quadro muda drasticamente ao interligar esses sistemas ou redes à Internet, pois o número de atacantes potenciais aumenta drasticamente. O conhecimento técnico dos atacantes varia em uma gama bem maior de áreas, aplicações e sistemas operacionais. A dificuldade de rastrear um ataque garante ao atacante uma relativa segurança de que não poderá ser responsabilizado pelos seus atos.

Antes de apresentar as falhas presentes nos três paradigmas, faz-se necessário categorizá-los conforme suas características de segurança. Em 1985, o Departamento de Defesa Americano apresentou o *Trusted Computer System Evaluation Criteria (TCSEC)*, uma classificação em quatro níveis de segurança a serem utilizados ao classificar sistemas operacionais quanto à segurança [DoD, 1985]. Entretanto, o ponto mais importante do documento com relação a este artigo é a classificação dos tipos de controle de acesso: *DAC* - Controle de Acesso Discricionário, e *MAC* - Controle de Acesso Mandatório. No primeiro, a segurança dos arquivos fica à discrição do usuário que o possui, e estes podem alterar as permissões de seus arquivos com relação a si próprio e a outros de forma livre. No segundo, existem políticas de segurança, comuns ou específicas a cada usuário, que restringem as ações que este pode realizar sobre o arquivo, visando não só a segurança do usuário, mas também a segurança de todo o sistema operacional.

Como pode ser observado, os três paradigmas apresentados na Seção 2 se enquadram na classificação *DAC*. Fica sob responsabilidade do usuário preocupar-se com a segurança de seus arquivos, através da configuração correta de permissões, ou através da cifragem dos documentos importantes. Mesmo sobre o super-usuário recai essa obrigação, dado o tipo de acesso que possui ao sistema operacional.

Com relação ao primeiro paradigma, existem dois pontos principais a serem analisados. É comum o usuário negligenciar a configuração correta das permissões sobre seus

arquivos, seja por desconhecimento de como fazê-lo, seja por descuido ou descaso do mesmo. Conseqüentemente, caso o domínio de outro usuário seja comprometido, e as permissões do primeiro usuário assim o permitirem, a partir do outro será possível obter acesso aos arquivos, ou até destruí-las. O segundo ponto está relacionado com o domínio associado às aplicações. Estas possuem acesso ao mesmo domínio do usuário que as executou, e, caso sejam comprometidas por um ataque bem sucedido, o invasor terá acesso ao mesmo domínio do usuário. Esse ponto tem maior implicação quando em conjunto com o segundo paradigma.

Em se tratando de um ataque a aplicações de usuários comuns, apenas os arquivos aos quais os mesmos possuem acesso poderão ser copiados e/ou destruídos. O risco maior encontra-se na possibilidade de que aplicações que necessitem de privilégios adicionais sejam comprometidas, pois, em função do segundo paradigma, uma vez que isso aconteça, todo o sistema estará comprometido. O problema, portanto, é a falta de granularidade na distribuição das operações privilegiadas do sistema operacional. Esta falta de granularidade decorre do fato de que todos os privilégios estão centralizados e isolados em um único domínio, o do super-usuário, o que impossibilita restringir a aplicação apenas ao subconjunto mínimo de operações privilegiadas necessárias.

Depender exclusivamente das aplicações para garantir a segurança é extremamente perigoso, como pode ser observado pelos incidentes de segurança reportados [CERT, 2003]. As aplicações estão se tornando cada vez mais complexas, possuem códigos cada vez maiores, e garantir a inexistência de falhas é praticamente impossível. Da identificação de uma vulnerabilidade até a disponibilização de uma correção gasta-se um tempo considerável, em que as aplicações atingidas podem ser exploradas por atacantes. Adicione-se o fato de que, com o acréscimo de novas funcionalidades, novas falhas podem surgir, e percebe-se claramente que continuar confiando nas aplicações é demasiado arriscado.

O terceiro paradigma apresentado pode ser considerado isoladamente seguro. Com chaves adequadas à segurança desejada para os arquivos, nenhum atacante poderá acessar o conteúdo. Entretanto, como apresentado anteriormente, o super-usuário pode acessar o conteúdo em memória durante o processo de cifragem e comprometer a segurança e, como visto, é possível a um atacante obter acesso ao domínio do super-usuário. Adicionalmente, mesmo que o conteúdo não possa ser lido, ele pode ser apagado, apenas com acesso ao domínio do usuário proprietário do arquivo.

Em [Loscocco et al., 1998] são apresentados argumentos que demonstram que, sem um suporte adequado de segurança por parte do sistema operacional, os problemas apresentados acima continuarão ocorrendo. Dessa forma, é possível então compreender porque os incidentes de segurança crescem a cada ano, e porque todo o esforço despendido em segurança de computadores tem se mostrado insuficiente para conter o avanço dos atacantes. Portanto, fazem-se necessários novos paradigmas de segurança para sistemas operacionais, que possam sustentar um desenvolvimento computacional seguro, sem depender exclusivamente das aplicações.

4. Novos Paradigmas de Segurança

A necessidade de que o sistema operacional garanta a segurança dos arquivos, aplicações e recursos do computador contra o eventual comprometimento de uma aplicação é inegável [Loscocco et al., 1998]. Partindo desse princípio, podem ser derivados três paradigmas de segurança: controle de acesso mandatório, princípio do privilégio mínimo e proteção dos mecanismos da *API* do sistema operacional.

O controle de acesso mandatório (*MAC*), como foi visto rapidamente na Seção 3, consiste em um conjunto de políticas de segurança que limitam as permissões que o usuário dispõe sobre arquivos e aplicações. Neste contexto, surge a necessidade de um administrador de políticas de segurança, distinto e isolado do administrador do sistema operacional, cuja responsabilidade é definir quais as políticas a serem aplicadas, visando evitar que os usuários, inclusive o super-usuário, exponham acidentalmente arquivos ou aplicações que possam comprometer o sistema operacional. A necessidade de isolamento entre o administrador de políticas e o administrador do sistema ocorre em virtude da falha apresentada pelo segundo paradigma da Seção 2. Caso contrário o comprometimento do super-usuário implicaria no comprometimento das políticas, e conseqüentemente, de todo o sistema operacional. Repare que, neste contexto, o super-usuário também se submete às políticas.

Uma necessidade proveniente do uso de políticas de segurança é a inclusão de dois módulos ao sistema operacional: o gerenciador de políticas (*policy manager*) e o restritor de segurança (*security enforcer* ou *reference monitor* [DoD, 1985]). O primeiro é responsável por gerenciar as políticas de segurança, utilizando uma linguagem inteligível aos administradores, e por converter as políticas de segurança para um formato inteligível ao restritor de segurança. O segundo deve garantir que as políticas sejam aplicadas, impedindo qualquer usuário, inclusive o administrador, de realizar alguma ação não autorizada. Adicionalmente, o sistema operacional deve garantir que o restritor seja sempre consultado, evitando uma tentativa de contorná-lo para subverter o sistema operacional.

O princípio do privilégio mínimo, apresentado em [Ferraiolo and Kuhn, 1992], consiste na idéia de que a um usuário ou aplicação não sejam dados nenhum privilégio a mais do que o necessário para realizar suas atividades. Como visto, o problema do comprometimento de uma aplicação consiste em que a mesma tem acesso ao mesmo domínio que o usuário que o está executando. Entretanto, ao restringir a aplicação a um subdomínio desse domínio, o estrago que poderá vir a ser causado será menor. Associando este paradigma ao paradigma de *MAC*, pode-se garantir que uma vulnerabilidade explorada nem sequer afete o usuário. Por exemplo, um navegador de Internet necessita de acesso de leitura as configurações do sistema operacional referentes a si próprio, de acesso de leitura e escrita a um diretório temporário e nada mais. Sendo possível conter a aplicação a esse subdomínio, mesmo que o navegador seja comprometido, o atacante somente terá acesso as configurações referentes ao navegador e ao diretório temporário, usado normalmente apenas para *cache* das páginas abertas pelo usuário. Mesmo que tentasse implantar um *trojan horse* no diretório temporário na esperança que o usuário o execute acidentalmente, uma política que impedisse a execução de aplicações no diretório temporário eliminaria esse risco.

Finalmente, o sistema operacional é responsável por proteger os mecanismos da

API contra ataques de *spoofing*, contorno (*bypassing*) e interferência (*tampering*). Muitas aplicações dependem dos mecanismos do sistema operacional para realizarem sua função corretamente, como canais de comunicação, mecanismos de criptografia, controle de acesso, acesso aos dispositivos de entrada e saída (teclados, mouse, vídeo), entre outros. Se os mesmos puderem ter seu comportamento alterado, seja por outra aplicação, seja por explorar alguma vulnerabilidade do sistema operacional, não há garantia de que as informações não estejam sendo destruídas ou registradas por terceiros. Mesmo a presença de outros métodos de segurança não serão suficientes para evitar a invasão.

Já existem vários sistemas operacionais que foram desenvolvidos com foco nos novos paradigmas de segurança. Destes, destacam-se os baseados em *micro-kernel*, que prevêem mecanismos primitivos de proteção e que visam suportar a construção de uma arquitetura de segurança flexível e de alto nível. Alguns apresentam uma limitação com relação ao suporte de controle de acesso mandatório com alto grau de flexibilidade e validação, como o Fluke [Ford et al., 1996] e o Exokernel [Mazieres and Kaashoek, 1997]. Outros, como o L4 [Liedtke, 1996], suportam controle de acesso mandatório com mecanismos de "clãs e chefes" e identificação de emissores/receptores na comunicação entre processos, porém são prejudicados pela falta de um arcabouço (*framework*) coerente para usar esses mecanismos com políticas mandatórias, e por preverem um pequeno número de domínios de segurança. Dois dos sistemas operacionais mais relevantes são o Flask (uma variação do Fluke [Secure Computing Corporation, 1998]) e o DTOS (uma variação do sistema operacional Mach). Ambos apresentam um arcabouço coerente com gerenciador de políticas e restritor de segurança [Loscocco et al., 1998].

Com relação às políticas de segurança, existem vários modelos que representam as relações entre usuários, aplicações e recursos. Um dos mais antigos, o *Multi-level Security* (*MLS* [Bell and Padula, 1973]), é baseado em decisões de liberação de usuários e classificações para objetos. Porém, ele é muito limitado para enquadrar vários requisitos de segurança. O *Domain and Type Enforcement* (*DTE* [Walker et al., 1996]), por sua vez, associa tipos aos arquivos e recursos do sistema e domínios às aplicações e usuários, sendo que os domínios definem qual o acesso permitido aos diferentes tipos. Outro modelo é o *Role-based Access Control* (*RBAC* [Ferraiolo and Kuhn, 1992]), que utiliza o conceito de papéis para definir as operações que podem ser realizados sobre os arquivos do sistema, de forma muito similar ao *DTE*. Finalmente, em [Swift et al., 2002] é apresentado um conjunto de extensões do modelo de *Access Control List* (*ACL*), que permite ao mecanismo de controle de acesso proteger o sistema, os recursos, as aplicação e os usuários, utilizado pelo Windows 2000.

5. Dificuldades da Implantação dos Novos Paradigmas de Segurança

Como foi visto na seção anterior, já existem paradigmas de segurança mais robustos que os presentes nos sistemas operacionais de uso comum, sob os quais foram desenvolvidos sistemas operacionais de nova geração, em que segurança é um fator primário. Entretanto, ao observar qualquer organização, será perceptível que os computadores de suas redes ainda utilizam largamente, se não totalmente, os sistemas operacionais baseados nos antigos paradigmas de segurança. Sendo a segurança uma preocupação crescente

em organizações [Nakamura and de Geus, 2002], faz-se necessário analisar os motivos da demora na adoção desses sistemas.

Um dos fatores que dificultam a implantação dos novos paradigmas é a "inércia" dos sistemas operacionais. Por "inércia", entenda-se resistência à migração. Seja por parte dos usuários, seja por causa das aplicações, seja devido a questões financeiras envolvendo licenças de sistemas operacionais, a sua substituição por outro não será fácil. Será focada a questão referente às aplicações, visto que é a que impõe maior complicação a longo prazo. Elas estão associadas às atividades das organizações, e dependem fortemente da estrutura a elas disponibilizada pelo sistema operacional para o qual foram desenvolvidas. A substituição do sistema operacional cria a necessidade de adaptação das aplicações ao novo sistema, ou ao desenvolvimento de novas aplicações que permitam às organizações continuarem a desempenhar suas atividades. Entretanto, ambos os processos envolvem altos custos financeiros e longo tempo de desenvolvimento e treinamento.

Sendo a substituição do sistema operacional inviável, a solução seria adaptar os sistemas operacionais existentes para se adequarem aos novos paradigmas. Entretanto, devido ao fato destes terem se desenvolvido a partir de modelos mais antigos, a estrutura não é própria para os novos modelos, requerendo uma reestruturação do sistema, com ênfase no núcleo, que deverá ser responsável por prover os dados necessários para a tomada de decisão do restritor de segurança e responder de acordo com a decisão. Outros sistemas auxiliares, como o sistema de arquivos, deverão ser remodelados, para poderem armazenar as informações adicionais (os tipos dos arquivos, as políticas relacionadas) necessárias à segurança. Neste sentido, já está em fase avançada de desenvolvimento um núcleo para o Linux com suporte a controle mandatório, o SELinux (*Security-Enhanced Linux*) [Loscocco and Smalley, 2001a, Loscocco and Smalley, 2001b] que possui um gerenciador de políticas com suporte a *DTE* e *RBAC*. O Windows também parece caminhar nesta direção, visto que já dispõe de controle de acesso de maior granularidade e mecanismos de *MAC* [Swift et al., 2002].

Outro fator que pesa na adoção dos novos paradigmas é a complexidade de gerenciamento de políticas de segurança. As políticas são de difícil compreensão, pouco escaláveis e limitadas em termos de usuários, funções, operações e classes [Baker, 1996]. Existem ferramentas próprias para auxiliar na identificação de conflitos entre políticas e na identificação de regiões não cobertas pelo conjunto de políticas (tanto restritivamente, quanto permissivelmente), como em [Jaeger et al., 2003], porém tais ferramentas somente auxiliam na verificação das políticas, e não em sua definição.

Para reduzir a complexidade de implementação do SELinux, algumas distribuições já disponibilizam o núcleo deste como opcional, e instruções de como instalá-lo [Gilburd, 2003], provendo inclusive algumas políticas básicas para os principais programas utilizados. Entretanto, o uso dessas políticas dificulta o trabalho do administrador de políticas, que primeiro precisa compreender o significado das milhares de linhas de definições pré-configuradas antes de poder alterá-las e estendê-las.

6. Considerações Finais

Como visto ao longo do artigo, os problemas de segurança em sistemas operacionais que ocorrem atualmente estão associados não somente às aplicações, mas também aos

paradigmas de segurança sobre os quais se baseiam os sistemas operacionais de uso mais comum, que não conseguem conter a falha somente à aplicação. Provenientes de uma geração onde os riscos eram bem menores e isolados, esses sistemas não dispõem da segurança necessária ao atual quadro computacional, cuja expansão das redes em caráter global propiciou um ambiente em que ataques são cada vez mais freqüentes. Entretanto, a substituição de sistemas operacionais não pode ser realizada rapidamente, em razão das aplicações já desenvolvidas e largamente utilizadas.

Para garantir a segurança dos sistemas operacionais faz-se necessário focar a atenção em novos paradigmas. Não que a segurança de redes deva ser relaxada ou esquecida, ou que as aplicações não devam passar por etapas de desenvolvimento visando a identificação e eliminação dos erros, pois mesmo isoladamente essas aplicações ainda podem causar problemas aos usuários. Neste sentido, as técnicas de programação segura devem ser aprimoradas e difundidas, e ainda há muito muito o que fazer em termos de topologias de redes, *firewalls*, entre outros.

O ponto chave deste artigo é que está na hora de se olhar com mais cuidado a segurança que pode ser obtida a partir dos sistemas operacionais, sem depender de aplicações auxiliares. É necessário que práticas como as que estão sendo realizadas no Linux para adequar-se aos novos paradigmas (o SELinux) sejam adotadas pelos outros sistemas operacionais em uso, reforçando as barreiras já existentes e limitando o acesso descontrolado obtido a partir de ataques bem sucedidos a aplicações.

Adicionalmente, as políticas de segurança devem ser estudadas, visando obter políticas mais claras e de fácil definição, para possibilitar a validação das mesmas. Ferramentas que simplifiquem o trabalho de definição e verificação de conflitos de políticas devem ser desenvolvidas e aprimoradas, objetivando não só o gerenciamento de forma compacta porém completa das políticas de segurança, mas também estendendo ao usuário a possibilidade de criar políticas adicionais para se adequar às suas necessidades.

Trabalhos futuros que devem ser realizados envolvem estudos mais aprofundados dos novos sistemas operacionais (incluindo nestes o SELinux), pesquisa de novas formas de representação e definição de políticas e a pesquisa e desenvolvimento de ferramentas que simplifiquem o gerenciamento de políticas. Dessa forma, será viabilizada a utilização dos novos paradigmas em larga escala.

Referências

Baker, D. (1996). Fortresses built upon sand. In *Proceedings of the New Security Paradigms Workshop*.

Bell, D. E. and Padula, L. J. L. (1973). Secure computer systems: Mathematical foundations and model. Technical report, The MITRE Corporation, Bedford, MA, EUA.

CERT (2003). Cert/cc statistics 1988-2003. Disponível em <http://www.cert.org/stats/>. Acesso em: 16/02/04.

DoD (1985). *Trusted Computer Security Evaluation Criteria*. Department of Defense. DOD 5200.28-STD.

Feiertag, R. J. and Organick, E. (1971). The multics input-output system. In *Proceedings of the Third Symposium on Operation Systems Principles*, pages 35–41, New York.

Ferraiolo, D. and Kuhn, R. (1992). Role-based access control. In *Poceedings of The 15th National Computer Security Conference*.

Ford, B., Hibler, M., Lepreau, J., abd Godmar Back, P. T., and Clawson, S. (1996). Microkernels meet recursive virtual machines. In *Proceedings of 2nd USENIX Symposium on Operating Systems Design and Impementation*.

Garfinkel, S. and Spafford, G. (1996). *Pratical Unix & Internet Security*. O'Reilly & Associates, Inc., USA, 2nd edition. 971 p.

Gilburd, Z. (2003). Gentoo x86 selinux installation guide. Disponível em: <http://www.gentoo.org/proj/en/hardened/selinux/selinux-x86-install.xml>.

Harrison, M. A., Ruzzo, W. L., and Ullman, J. D. (1976). Protection in operating systems. *Commun. ACM*, 19(8):461–471.

Jaeger, T., Zhang, X., and Cacheda, F. (2003). Policy management using access control spaces. *ACM Trans. Inf. Syst. Secur.*, 6(3):327–364.

Liedtke, J. (1996). *L4 Reference Manual*. IBM T. J. Watson Research Center. RC 20549.

Loscocco, P. and Smalley, S. (2001a). Integrating flexible support for security policies into the linux operating system. In *Proccedings of the FREENIX Track: 2001 USENIX Annual Technical Conference*, Boston Mass.

Loscocco, P. and Smalley, S. (2001b). Meeting critical security objectives with security-enhanced linux. In *Proccedings of The 2001 Ottawa Linux Symposium*, Ottawa, Ont., Canada.

Loscocco, P. A., Smalley, S. D., Muckelbauer, P. A., Taylor, R. C., Turner, S. J., and Farrel, J. F. (1998). The inevitability of failure: The flawed assumption of security in modern computing environment. In *Proceedings of the 21st National Information Systems Security Conference*, pages 303–314.

Mazieres, D. and Kaashoek, M. (1997). Secure applications need flexible operating systems. In *Proceedings of the 6th Workshop on Hot Topics in Operating Systems*.

Nakamura, E. and de Geus, P. L. (2002). *Segurança de Redes em ambientes cooperativos*. Editora Berkeley, São Paulo, first edition.

Ritchie, D. M. and Thompson, K. (1974). The unix time-sharing system. *Commun. ACM*, 17(7):365–375.

Secure Computing Corporation (1998). Assurance in the fluke microkernel: Formal security model. Technical report, Secure Computing Corporation. MD A904-97-C-3047 CDRL A003.

Silbertschatz, P. B., Galvin, P. B., and Gagne, G. (2002). *Operating System Concepts*. John Wiley & Sons, Inc., New York, 6th edition. 887 p.

Swift, M. M., Hopkins, A., Brundrett, P., Van Dyke, C., Garg, P., Chan, S., Goertzel, M., and Jensenworth, G. (2002). Improving the granularity of access control for windows 2000. *ACM Trans. Inf. Syst. Secur.*, 5(4):398–437.

Tanenbaum, A. S. (2003). *Computer Networks*. Prentiche Hall, New Jersey - USA, 4th edition.

Tanenbaum, A. S. and Woodhull, A. S. (1997). *Operating systems: Design and Implementation*. Prentice Hall, New Jersey - USA, 2nd edition.

Walker, K. W., Bagder, D. F., Petkac, M. J., Sherman, L., and Oostendorp, K. A. (1996). Confining root programs with domain and type enforcement. In *Proceedings of The 6th USENIX Security Symposium*, San Jose, California.