Detecção de intrusão em máquinas virtuais

ARTICLE · JANUARY 2003

READS
281

4 AUTHORS, INCLUDING:

Carlos Maziero
Universidade Federal do Paraná
72 PUBLICATIONS **209** CITATIONS
SEE PROFILE

Edgard Jamhour
Pontifícia Universidade Católica do Paraná …
60 PUBLICATIONS **165** CITATIONS
SEE PROFILE

Anais do 5º SSI – Simpósio de Segurança em Informática – São José dos Campos – SP, 2003.

DETECÇÃO DE INTRUSÃO EM MÁQUINAS VIRTUAIS

Marcos Aurelio Pchek Laureano, Carlos Alberto Maziero, Edgard Jamhour
Programa de Pós-Graduação em Informática Aplicada
Pontifícia Universidade Católica do Paraná
80215-901 Curitiba – PR
{laureano,maziero,jamhour}@ppgia.pucpr.br

RESUMO

A utilização de máquinas virtuais está se tornando uma alternativa para vários sistemas de computação, pelas vantagens em custos e portabilidade. O conceito de máquina virtual também pode ser empregado para melhorar a segurança de um sistema computacional contras ataques a seus serviços. Este artigo apresenta e avalia uma nova abordagem para aumentar a segurança de sistemas, aplicando técnicas de detecção de intrusão em ambientes baseados em máquinas virtuais, para detectar e combater ataques a serviços de rede.

ABSTRACT

Virtual Machines are becoming valuable in production computing systems, because of their benefits in terms of costs and portability. The virtual machine concept can also be used to improve the security of a computer system in face of attacks to its network services. This paper presents a new approach to achieve this goal, by applying intrusion detection techniques to virtual machine – based systems.

1. INTRODUÇÃO

O conceito de máquina virtual não é novo – suas origens remetem ao início da história dos computadores, no final dos anos 50 e anos 60 (Goldberg, 1973; Varian, 1989). A utilização de máquinas virtuais está se tornando uma alternativa para vários sistemas de computação, pelas vantagens em custos e portabilidade (Blunden, 2002; Silberchatz e Galvin, 2000), inclusive em sistemas de segurança (Honeynet Project, 2003). Um uso freqüente de sistemas baseados em máquinas virtuais é a chamada "consolidação de servidores": em vez da utilização de vários equipamentos com seus respectivos sistemas operacionais, utiliza-se somente um computador, com máquinas virtuais abrigando vários sistemas operacionais e suas respectivas aplicações e serviços.

O conceito de máquina virtual pode ser empregado para melhorar a segurança de um sistema computacional contras ataques a seus serviços. Um problema central em segurança de sistemas é a dificuldade em obter dados confiáveis de um sistema invadido. Uma vez ocorrida uma invasão, normalmente não é possível extrair informação confiável do sistema em funcionamento, pois suas ferramentas de monitoração e gerência podem ter sido adulteradas para ocultar a presença do invasor.

O presente trabalho apresenta uma proposta para aumentar a confiabilidade de sistemas nesse contexto, aplicando mecanismos de detecção de intrusão em ambientes de máquinas virtuais, para detectar e combater ataques a serviços de rede. Este artigo está assim estruturado: o capítulo 2 apresenta os principais conceitos relacionados a máquinas virtuais conforme empregados neste trabalho; o capítulo 3 conceitua e detalha aspectos de segurança e abordagens para detecção de intrusão; o capítulo 4 detalha a proposta do artigo e apresenta os resultados obtidos até o momento.

2. MÁQUINAS VIRTUAIS

Uma máquina virtual (*Virtual Machine* - VM) é definida em (Popek e Goldberg, 1974) como "uma duplicata eficiente e isolada de uma máquina real". Em uma máquina real, uma camada de software de baixo nível (por exemplo, a BIOS dos sistemas PC) fornece acesso aos vários recursos do hardware para o sistema operacional, que os disponibiliza de forma abstrata às aplicações. Em uma máquina real, quando o sistema operacional acessa os dispositivos de hardware, ele faz uso dos *device drivers* respectivos, que interagem diretamente com a memória e os dispositivos de E/S da máquina.

Um *emulador* é o oposto da máquina real. O emulador implementa todas as instruções realizadas pela máquina real em um ambiente abstrato, possibilitando executar um aplicativo de uma plataforma em outra, por exemplo, um aplicativo do Windows executando no Linux. Infelizmente, um emulador perde muito em eficiência ao traduzir cada instrução da máquina real. Além disso, emuladores são bastante complexos, pois geralmente necessitam simular a quase totalidade das instruções do processador e demais características do hardware que os circundam (Mallach, 1973).

A funcionalidade e o nível de abstração de uma máquina virtual encontra-se entre uma máquina real e um emulador, na medida em que abstrai somente os recursos de hardware e de controle usados pelas aplicações. Uma máquina virtual é um ambiente criado por um monitor de máquinas virtuais (*Virtual*

Machine Monitor – VMM), também denominado "sistema operacional para sistemas operacionais" (Kelem e Feiertag, 1991). O VMM pode criar uma ou mais máquinas virtuais sobre uma única máquina real. Enquanto um emulador fornece uma camada de abstração completa entre o sistema em execução e o hardware, um VMM abstrai o hardware subjacente e controla uma ou mais máquinas virtuais. Cada VM fornece facilidades para uma aplicação ou um "sistema convidado" que acredita estar executando sobre um ambiente normal de acesso físico ao hardware.

Existem basicamente duas abordagens para a construção de sistemas de máquinas virtuais: o tipo I (figura 1), onde o monitor de máquinas virtuais é implementado entre o hardware e os sistemas convidados, e o tipo II (figura 2), onde o monitor é implementado como um processo de um sistema operacional real subjacente, denominado *sistema anfitrião*.

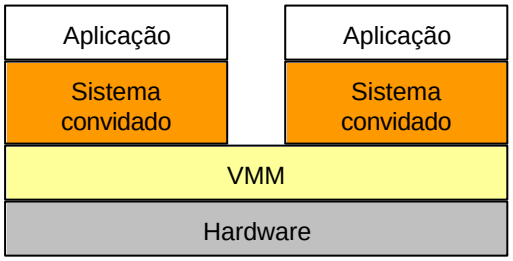

Figura 1: Monitor de máquinas virtuais de tipo I

Este artigo tem como pressuposto a aplicação de máquinas virtuais de tipo II na segurança de sistemas. Como o sistema operacional convidado o e ambiente de execução na máquina virtual são idênticos ao da máquina real, é possível usar os softwares já construídos para a máquina real dentro das máquinas virtuais. Essa transparência evita ter de construir novas aplicações ou adaptar as já existentes.

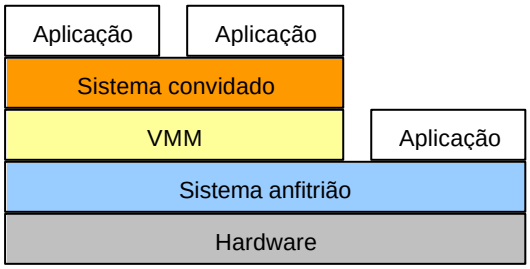

Figura 2: Monitor de máquinas virtuais de tipo II

As máquinas virtuais têm recebido bastante destaque nos últimos anos, sobretudo devido ao grande sucesso de linguagens independentes de plataforma como Java. Todavia, a máquina virtual Java (*Java Virtual Machine* – JVM) deve ser considerada basicamente um emulador, na medida em que cria um ambiente de execução abstrato para aplicações Java. Programas em linguagem Java são compilados em *bytecodes*, que são basicamente código de máquina para um processador abstrato. Assim, a JVM somente executa aplicações construídas especificamente para esse ambiente (Esteire e Lovelle, 1998).

2.1 *Razões de Uso de Máquinas Virtuais*

Em (Goldberg, 1973; Özden, Goldberg e Silberschatz, 1994; Agren, 1999; Silberchatz e Galvin, 2000; Blunden, 2002) são relacionados algumas vantagens para a utilização de máquinas virtuais em sistemas de computação:

- Aperfeiçoamento e testes de novos sistemas operacionais;
- Ensino prático de sistemas operacionais e programação;
- Executar diferentes sistemas operacionais sobre o mesmo hardware, simultaneamente;
- Simular configurações e situações diferentes do mundo real, como por exemplo, mais memória disponível, outros dispositivos de E/S;
- Simular alterações e falhas no hardware para testes ou re-configuração de um sistema operacional, provendo confiabilidade e escalabilidade para as aplicações;
- Garantir a portabilidade das aplicações legadas (que executariam sobre uma VM simulando o sistema operacional original);
- Desenvolvimento de novas aplicações para diversas plataformas, garantindo a portabilidade destas aplicações;
- Diminuir custos com hardware.

A principal desvantagem do uso de máquinas virtuais é o custo adicional de execução dos processos na máquina virtual em comparação com a máquina real. Esse custo é muito variável, podendo chegar a 50% ou mais em plataformas sem suporte de hardware a virtualização, como os PCs de plataforma Intel (Dike, 2000; Blunden, 2002; VMWare Inc., 1999; Whitaker, Shaw e Gribble, 2002). Todavia, pesquisas recentes têm obtido a redução desse custo a patamares abaixo de 20%, graças sobretudo a ajustes no código do sistema anfitrião (Whitaker, Shaw e Gribble, 2002; King e Chen, 2002; King, Dunlap e Chen, 2003). Esse problema não existe em ambiente cujo hardware suporta o conceito de virtualização, como é o caso de *mainframes*.

3. SEGURANÇA DE SISTEMAS

A segurança da informação é um conjunto de medidas que se constituem basicamente de controles (através de ferramentas) e políticas, tendo como objetivo a proteção das informações dos clientes e da empresa, controlando o risco de revelação ou alteração por pessoas não autorizadas.

O ataque é ato de tentar desviar dos controles de segurança de um sistema. Um ataque pode ser *ativo*, tendo por resultado a alteração dos dados; *passivo*, tendo por resultado a liberação dos dados; ou *destrutivo* visando à negação do acesso aos dados ou serviços (Wadlow, 2000). O fato de um ataque estar acontecendo não significa necessariamente que ele terá sucesso. O nível de sucesso depende da vulnerabilidade do sistema ou da atividade e da eficácia de contramedidas existentes (Stallings, 1998; Wadlow, 2000).

Um atacante pode ter vários objetivos ao realizar um ataque, como causar prejuízo ao atacado ou procurar vulnerabilidades no sistema para depois divulgar estas informações. Estas vulnerabilidades podem estar ligadas a um sistema mal-configurado ou a falhas na construção de um software, como as falhas que possibilitam ataques de *buffer overflow* e *format string*.

São vários os tipos de ataques existentes, com os mais variados objetivos (roubo de informação, negação de serviço, invasão para facilitar outros ataques). Alguns exemplos de ataques:

- *Ataque dirigido a dados* – Forma de ataque em que o ataque é codificado nos dados normais e inócuos que são executados por um usuário ou por outro software.
- *Negação de Serviço* – O impedimento do acesso autorizado aos recursos ou o retardamento de operações críticas por tempo.
- *Abuso de privilégio* – Quando um usuário realiza uma ação que ele não estava autorizado a realizar; uso impróprio ou mal intencionado dos recursos da tecnologia da informação.
- *Fraude* – Crimes relacionados a computadores envolvendo deliberada alteração, falsificação ou divulgação de dados com a intenção de obter algo de valor (normalmente ganho monetário). Um sistema de computador deve ter sido envolvido na realização ou no encobrimento do ato ou série de atos fraudulentos.
- *Vírus* – Uma classe do software malicioso que tem a habilidade de se auto-replicar e infectar partes do sistema operacional ou dos programas de aplicação, com o intuito de causar a perda ou dano nos dados.

São várias as formas de se combater um ataque (Wadlow, 2000), entre as quais estão a utilização de *firewalls*, utilização de antivírus com constantes atualizações, instalação de correções de segurança, políticas de segurança bem documentadas e acessíveis a todos os funcionários, treinamento adequado, configurações corretas e boa administração dos sistemas operacionais, utilização de criptografia e certificação digital para a proteção dos dados e mecanismos de detecção de intrusão.

4. DETECÇÃO DE INTRUSÃO

Nos últimos anos, uma tecnologia tem se mostrado uma grande aliada dos administradores de segurança. São os Sistemas de Detecção de Intrusão (IDS - *Intrusion Detection System*). Basicamente, o que tais sistemas fazem é tentar reconhecer um comportamento ou uma ação intrusiva para alertar um administrador ou automaticamente disparar contra-medidas.

O IDS automatiza a tarefa de analisar estes dados da auditoria. Estes dados são extremamente úteis, pois podem ser usados para estabelecer a culpabilidade do atacante e na maioria das vezes é o único modo de descobrir uma atividade sem autorização, detectar a extensão dos danos e prevenir tal ataque no futuro, tornando desta forma o IDS uma ferramenta valiosa para análises em tempo real e também após a ocorrência de um ataque.

Existem duas abordagens para a implementação de ferramentas IDS:

- *Host Based* – nesta abordagem monitora-se a atividade local em uma máquina, como processos, conexões de rede, chamadas de sistema e níveis de uso de recursos locais, com o objetivo de identificar ataques e tentativas de acesso indevido à própria máquina. Neste caso, o IDS é instalado na máquina a monitorar.
- *Network Based* – baseia-se na captura e análise do tráfego de rede, buscando detectar assinaturas de ataques conhecidos e/ou padrões anormais de atividade em máquinas da rede monitorada. O IDS deve ser instalado em uma máquina que tenha visibilidade total sobre o tráfego de rede a monitorar.

Os sistemas de detecção de intrusão baseados em rede são mais abrangentes, pois podem monitorar diversos computadores simultaneamente. Todavia, sua eficácia diminui na medida em que o tamanho e a velocidade da rede aumenta, pela necessidade de analisar os pacotes mais rapidamente. Além disso, o uso cada vez maior de protocolos cifrados (baseados em SSL) torna o conteúdo dos pacotes opaco ao IDS.

A velocidade da rede e o uso de criptografia não são problemas para os sistemas de detecção de intrusão baseados em *host*. Todavia, como esse sistema é instalado na própria máquina a monitorar, pode ser desativado após um ataque bem-sucedido. O presente trabalho apresenta uma proposta para solucionar esse problema.

As técnicas utilizadas na detecção de intrusão podem ser caracterizadas como detecção de anomalias e detecção de mau uso. A descoberta de um ataque bem-sucedido ou não, depende da análise do que está ocorrendo na rede. Pode-se utilizar *sniffers* que ficam "escutando" o tráfego de dados pela rede e comparando com um registro de assinaturas de possíveis ataques. Outros métodos de identificação analisam o comportamento dos sistemas, quando ocorre uma anomalia em relação

ao comportamento usual previamente registrado, o sistema de identificação emite um alerta. Neste caso, um comportamento não usual pode ser um comportamento válido (gerando uma mensagem de identificação falsa) e que deve ser registrado para usos futuros.

Em um dos trabalhos de referência nesta área, Dorothy Denning define em (Denning, 1987) quatro modelos estatísticos que podem ser utilizados em um detector de intrusão por anomalia. Cada modelo descrito na seqüência é considerado apropriado para um tipo particular de métrica.

- *Modelo operacional* – este modelo aplica-se a métricas como, por exemplo, contadores de eventos para o número de falhas de *login* em um determinado intervalo de tempo. O modelo compara a métrica a um limiar definido, identificando uma anomalia quando a métrica excede o valor limite. Esse modelo pode ser aplicado tanto na detecção por anomalia quanto na detecção por mau uso.
- *Modelo de média e desvio padrão* – este modelo propõe uma caracterização clássica de média e desvio padrão para os dados. Uma nova observação de comportamento é identificada como anormal se ela encontra-se fora de um intervalo de confiança. Esse intervalo de confiança é definido como sendo *d* desvios padrões da média, para algum parâmetro *d*. Denning levanta a hipótese de que essa caracterização é aplicável a métricas do tipo contadores de eventos, intervalos de tempo e medidas de recursos.
- *Modelo multivalorado* – este modelo é uma extensão ao modelo de média e desvio padrão. Ela é baseada na correlação entre duas ou mais métricas. Desse modo, ao invés de basear a detecção de uma anomalia estritamente em uma métrica, essa detecção é baseada na correlação dessa métrica com alguma outra medida.
- *Modelo de processo de Markov* – este modelo é mais complexo e limitado a contadores de eventos. Segundo o mesmo, o detector considera cada tipo diferente de evento de auditoria como uma variável de estado e usa uma matriz de transição de estados para caracterizar as freqüências com que ocorrem as transições entre os estados. Uma nova observação de comportamento é definida como anormal se sua probabilidade, determinada pelo estado anterior e pelo valor na matriz de transição de estados, for muito baixa. Esse modelo permite que o detector identifique seqüências não usuais de comandos e eventos, introduzindo a noção de análise de fluxos de eventos com memória de estado.

A detecção de intrusão pode ser considerada como um processo de análise de dados, onde a detecção de anomalias trata da identificação dos padrões de utilização normal, e classificação dos desvios de comportamento como ameaças (um dos modelos propostos por Denning), a partir de dados auditados; e a detecção de mau uso trata da identificação de padrões de ataques conhecidos, a partir de dados auditados.

5. DETECÇÃO DE INTRUSÃO EM MÁQUINAS VIRTUAIS

A garantia da segurança de execução de um sistema torna-se uma tarefa cada vez mais complexa, devido ao crescimento das formas de ataque e atacantes (Allen et all, 1999). Os sistemas de detecção de intrusão atuais apresentam várias deficiências, conforme apresentado na seção anterior.

O principal problema relacionado à detecção de intrusão baseada em *host* reside no fato que o IDS reside na máquina a monitorar, ou seja, *o sistema também deve monitorar a si próprio*. Assim, um IDS baseado em *host* torna-se inútil no caso de um ataque bem sucedido, pois as informações oriundas deste sistema não podem mais ser consideradas confiáveis (Wadlow, 2000).

O uso de máquinas virtuais permite solucionar esse problema, constituindo uma alternativa interessante para a implantação de sistemas detectores de intrusão baseados em *host*, conforme apresentado na proposta a seguir.

5.1 Proposta

A proposta deste trabalho é definir uma arquitetura para a aplicação de detectores de intrusão baseados em *host* de forma robusta e confiável. Isso é obtido através da execução dos processos de aplicação a monitorar em máquinas virtuais e a implantação dos sistemas de detecção e resposta fora da máquina virtual (e portanto fora do alcance de eventuais invasores). A figura 3 ilustra os principais componentes da arquitetura.

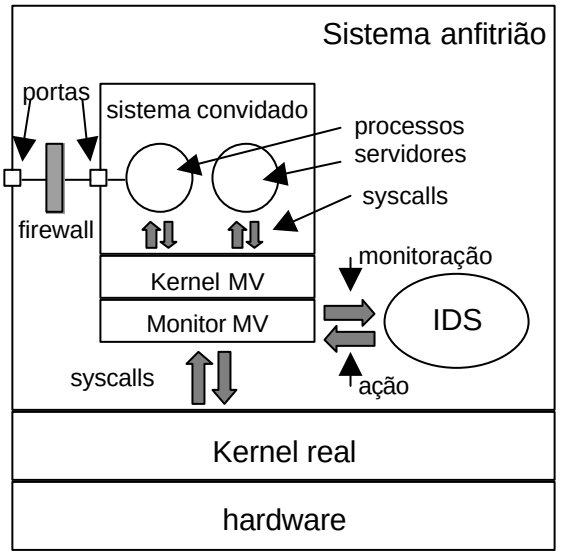

Figura 3: arquitetura proposta.

A interação do sistema convidado com a rede é feita através de um *firewall* de software, gerenciado pelo sistema real. Sob a ótica do sistema convidado, trata-se de um *firewall* externo e portanto inacessível aos invasores.

Devem ser definidos mecanismos para a interação entre o sistema convidado e os sistemas de detecção/ação externos. A interação entre o sistema real e o sistema convidado está dividida em duas partes:

- *Monitoração*: dados do sistema convidado são obtidos do monitor de máquina virtual para análise externa, como exemplo, podem ser disponibilizadas as chamadas de sistema dos processos convidados.
- *Ação*: a possibilidade de comandar externamente ações sobre o sistema convidado, como encerrar processos, remover arquivos, fechar conexões de rede, etc, em resposta a ataques detectados.

Além da ação sobre o sistema convidado, o sistema externo pode também agir sobre o *firewall* que liga o sistema convidado à rede externa, bloqueando portas e conexões conforme necessário.

A arquitetura apresentada torna o sistema de detecção/resposta inacessível a eventuais invasores. Fazendo a analogia com um sistema convencional, é como se o sistema de detecção/resposta tivesse sido implementado no hardware da máquina, tornando-o incontornável aos processos de usuário.

Para garantir a segurança do sistema é importante observar que:

- As interações com o sistema convidado devem ser feitas sempre através do monitor de máquina virtual;
- O monitor de máquina virtual deve ser inacessível aos processos de usuário do sistema convidado;
- Todos os serviços de rede devem ser providos pelos processos do sistema convidado. O acesso ao sistema real subjacente via rede deve ser evitado.

5.2 Implementação

Um protótipo da proposta foi implementado em plataforma Linux, usando o suporte para máquinas virtuais UML (*User-Mode Linux*) (Dike, 2000). O monitor de máquina virtual do UML foi modificado para permitir extrair informações sobre os processos executados pelo sistema convidado. Para a detecção de intrusão foi adotada uma abordagem por anomalia, baseada na análise das seqüências de chamadas de sistema dos processos convidados.

Na máquina real, foi implementado um programa, atualmente denominado VMIDS (*Virtual Machine Intrusion Detection System*), responsável pelo registro e análise das informações enviadas pelo monitor de máquina virtual. Além de monitorar as chamadas de sistema efetuadas pelos processos do sistema convidado, o VMIDS também permite definir os usuários autorizados a utilizar a máquina virtual e processos autorizados a executar na máquina virtual.

O sistema VMIDS pode operar em dois modos: *monitoração* e *registro*. Quando o VMIDS está sendo executado no modo *registro*, pode registrar os processos em execução e seus usuários como processos e usuários autorizados. Pode-se também registrar o fluxo de chamadas de sistema de processos específicos. O modo *registro* permite, portanto, registrar o comportamento "normal" do sistema, coletando dados que serão essenciais para a detecção de intrusão.

No modo monitoração, o VMIDS recebe informações do monitor de máquina virtual e as compara com os dados previamente armazenados. Caso uma seqüência de chamadas de sistema não esteja registrada para aquele processo, ou se o usuário que está executando o processo não está autorizado a executá-lo, ou ainda, se um processo não está autorizado a executar na máquina virtual, uma situação anormal é detectada pelo VMIDS.

O protótipo atual do VMIDS trabalha de forma passiva, ou seja, ele somente realiza a monitoração das chamadas, processos ou usuários autorizados e emite mensagens de alerta, não tomando nenhuma ação de prevenção ou correção das irregularidades encontradas. Estão previstas modificações para permitir a interação com o núcleo do sistema convidado, para agir sobre os processos do mesmo.

Foi utilizada a versão do *User-Mode Linux* com o *kernel* 2.4.20 para as referidas alterações. A comunicação entre o monitor de máquina virtual do UML e o processo VMIDS está sendo feita através de *named pipes*. Dessa forma, o próprio sistema operacional da máquina real gerencia o fluxo de informações entre os processos.

5.3 Resultados obtidos

Utilizando o sistema de arquivos de uma distribuição Linux Debian 2.2, foram efetuadas medidas de tempo para medir o impacto sobre os processos mais comuns de um sistema linux. Para realizar as medidas, foi utilizado o resultado do comando `time` da própria máquina virtual Linux. O equipamento utilizado nos testes foi um Athlon XP 1600, com 512 MB de memória, 30 GB de disco executando sobre a versão 8.2 do *Suse Linux Professional*.

Foram tomados os tempos de execução dos comandos `find`, `ls` e `ps` e feita a comparação das execuções da máquina real e da máquina virtual. A máquina virtual foi executada em três situações distintas: sem alterações (para referência), com as alterações e o VMIDS executando no modo registro e com as alterações e o VMIDS executando no modo monitor. Os parâmetros utilizados para compilação e execução das instâncias das máquinas virtuais foram às mesmas.

A tabela 1 apresenta os tempos médios de execução de cada comando e seus respectivos desvios-padrão. Pode-se observar que o tempo médio dos comandos, quando executados dentro do ambiente virtual, são superiores aos obtidos diretamente sobre a máquina real. Isso demonstra o custo computacional envolvido na virtualização, comentado na seção 2.1. A tabela 2 apresenta o custo ocasionado das alterações na máquina virtual para responder ao sistema de registro e monitoração.

Comando	Máquina Real		Máquina Virtual Original		Máq. Virtual VMIDS em Registro		Máq. Virtual VMIDS em Monitoração	
	T	Desvio	T	Desvio	T	Desvio	T	Desvio
`ps –ef`	0,022 s	7,78%	0,110 s	2,20%	0,127 s	6,26%	0,117 s	7,78%
`find / >/dev/null 2>&1`	0,016 s	2,53%	0,354 s	0,46%	0,511 s	1,00%	0,970 s	2,53%
`ls -laR / >/dev/null 2>&1`	0,058 s	2,28%	0,510 s	3,46%	0,745 s	1,03%	0,968 s	2,28%

Tabela 1: Tempo médio de execução.

Comando	Máq. Virtual Original (em relação à máquina real)	Máq. Virtual com VMIDS em modo Registro (em relação à máquina virtual)	Máq. Virtual com VMIDS em modo Monitoração (em relação à máquina virtual)
`ps –ef`	**489,29%**	**15,69%**	**6,16%**
`find / >/dev/null 2>&1`	**2243,04%**	**44,24%**	**73,65%**
`ls -laR / >/dev/null 2>&1`	**876,63%**	**45,98%**	**89,73%**

Tabela 2: Acréscimo de Tempo.

O comando `ps`, por efetuar poucas chamadas de sistema, levou um tempo maior para registrar (0,127 s) que para monitorar (0,117 s) por causa do custo ocasionado no sistema da máquina real no momento da gravação dos dados. Nos comandos `ls` e `find`, como a quantidade de chamadas de sistema é bem maior, o tempo para consulta à base de dados para encontrar eventuais padrões tornou maior o tempo de execução.

Foram realizadas testes preliminares de detecção de intrusão com versões alteradas (oriundas de *rootkits*) de alguns comandos do próprio sistema Linux, por exemplo, o comando `ps` que é alterado para não mostrar em sua relação determinados processos, o comando `login` e `su` que são alterados para registrar em arquivos os usuários e suas respectivas senhas. O sistema respondeu positivamente em 100% dos testes realizados.

5.4 Trabalhos Futuros

O VMIDS, embora seja funcional, ainda não está com um desempenho adequado ser utilizado em sistemas de produção. Atualmente estamos trabalhando no sentido de melhorar o desempenho da análise das informações, trabalhando com bancos de dados em vez de arquivos textos e registros em memória.

Como VMIDS é passivo, estamos estudando formas de ação no caso de detecção de situações anômalas. O principal objetivo é impedir a continuidade da execução do processo suspeito na máquina virtual. Pode-se interagir com o núcleo do sistema convidado para terminar ou suspender os processos suspeitos, ou então agir sobre o *firewall* da máquina real para bloquear as conexões de rede envolvidas.

Outros aspectos em estudo dizem respeito a implementar formas de monitoração baseadas em outras informações, como a análise de fluxo de rede da máquina virtual, a alocação de memória e o comportamento dos usuários sobre determinados processos. Algoritmos mais sofisticados de detecção de intrusão podem ser implementados a partir dessas informações, auxiliando a reduzir a ocorrência de resultados falsos (positivos e negativos).

5.5 Trabalhos Relacionados

Em (Chen e Noble, 2001) são citados alguns benefícios que a alteração dos códigos das máquinas virtuais podem trazer para a segurança e compatibilidade de sistemas, como a captura de mensagens de *log*, sistema de detecção de intrusão agindo sobre a máquina virtual (através do controle de estados) ou um ambiente de migração de sistemas, mas o artigo não demonstra como devem ser realizadas estas alterações, nem analisa seu impacto sobre o sistema.

O artigo (Dunlap, King, Cinar, Basrai e Chen, 2002) descreve outra experiência de uso de máquinas virtuais para a segurança de sistemas. A proposta prevê uma camada intermediária entre o VMM e o sistema anfitrião. Esta camada, chamada de *Revirt*, é responsável pela captura dos dados enviados através do *syslog* (*daemon* padrão Unix que registra as informações enviadas pelas aplicações em execução) da máquina virtual e seu envio para registro no sistema anfitrião. Essa abordagem faz uma cópia de todas as informações

enviadas pelas aplicações ao sistema virtual para análise posterior no sistema real, através de métodos tradicionais. Entretanto, caso o sistema virtual seja atacado e subvertido, as mensagens capturadas podem estar sendo manipuladas pelo atacante e conseqüentemente não serão mais confiáveis.

6. CONCLUSÃO

Este artigo descreve uma proposta para aumentar a segurança de sistemas computacionais através do uso de máquinas virtuais. A base da proposta é monitorar as ações dos processos da máquina virtual através de sistemas de detecção de intrusão externos à mesma. Os dados usados para a detecção de intrusão são obtidos por interação entre o monitor de máquina virtual e o processo que implementa o IDS na máquina real. Com isso, o sistema de detecção torna-se inacessível aos processos da máquina virtual e não pode ser subvertido por um eventual invasor.

O protótipo construído demonstra que a abordagem é viável e apresenta um desempenho satisfatório. Todavia, trabalhos complementares devem ser realizados para diminuir o custo de virtualização (uso de máquinas virtuais) e para melhorar o desempenho do mecanismo de detecção de intrusão implementado.

REFERÊNCIAS BIBLIOGRÁFICAS

AGREN, Ola. *Teaching Computer Concepts Using Virtual Machines*. SIGCSE Bulletin. Volume 31, Páginas 84-85. Junho de 1999.

ALLEN, Julia. et al. *State of the Practice of Intrusion Detection Technologies*. Technical Report CMU/SEI-99-TR028. Carnegie Mellon University. Janeiro de 2003.

BLUNDEN, Bill. *Virtual Machine Design and Implementation in C/C++*. Wordware Publishing, 2002. Plano, Texas – USA.

CHEN, Peter M. e NOBLE, Brian D. *When Virtual Is Better Than Real*. Proceedings of the 2001 Workshop on Hot Topics in Operating Systems (HotOS), 2001.

DENNING, Dorothy E. *An Intrusion-Detection Model*. IEEE Transactions on Software Engineering. Volume SE-13, Nº 02, Páginas 222-232. Fevereiro de 1987.

DIKE, Jeff. *A User-mode port of the Linux Kernel*. Proceedings of the 4ª Annual Linux Showcase & Conference, 2000. Atlanta – USA.

DUNLAP, George W., KING, Samuel T., CINAR, Sukru., BASRAI, Murtaza A. e CHEN, Peter M. ReVirt: *Enabling Intrusion Analysis through Virtual-Machine Logging and Replay*. Proceedings of the 2002 Symposium on Operating Systems Design and Implementation (OSDI), 2002.

ESTEIRE, Oscar Azanón. LOVELLE, Juan Manuel Cueva. *Set of tools for native code generation for the Java virtual machines***.** ACM SIGPLAN Notices, Volume 33, Página 73-79. Março de 1998.

GOLDBERG, R. P. *Architecture of Virtual Machines*. AFIPS National Computer Conference, 1973. New York – NY – USA.

HONEYNET PROJECT. *Know Your Enemy: Defining Virtual Honeynets – Differents types of Virtual Honeynets*. URL http://project.honeynet.org/papers/virtual. Janeiro de 2003.

KING, Samuel T. e CHEN, Peter M. *Operating System Extensions to Support Host Based Virtual Machines*. Technical Report CSE-TR-465-02, University of Michigan, Setembro de 2002.

KING, Samuel T., DUNLAP, George W. e CHEN, Peter M. *Operating System Support for Virtual Machines*. Proceedings of the 2003 USENIX Technical Conference.

KELEM, Nancy L. e FEIERTAG, Richard J. *A Separation Model for Virtual Machine Monitors. Research in Security and Privacy*. IEEE Computer Society Symposium, 1991, págs. 78-86.

MALLACH, Efrem G. *On the Relationship Between Virtual Machines and Emulators*. Workshop on Virtual Computer Systems, 1973. Cambridge – Massachusetts – USA. Páginas 117-126.

ÖZDEN, Banu. GOLDBERG, Aaron J. e SILBERSCHATZ, Avi. *Virtual Computers – A New Paradigm for Distributed Operating Systems*. AT&T Bell Laboratories – Murray Hill – New Jersey – USA. 1994

POPEK, Gerald J. e GOLDBERG, Robert P. *Formal Requirements for Virtualizable Third Generation Architectures*. Communications of the ACM. Volume 17, Número 7, Páginas 412-421, Julho de 1974.

SILBERCHATZ, Abraham e GA LVIN, Peter Baer. *Sistemas Operacionais: Conceitos.* Prentice Hall, 2000. São Paulo – SP.

STALLINGS, William. *Cryptography and Network Security: Principles and Practice*. Prentice-Hall, 1998 – 2ª Edição.

VARIAN, Melinda. *VM and the VM Community: Past, Present, and Future*. Sessions of SHARE. Sessions 9059-9061. Melbourne – Australia, 1989 (última revisão agosto de 1997).

VMware Inc. *VMware Technical White Paper*. Palo Alto – CA - USA Fevereiro 1999.

WADLOW, Thomas A. *Segurança de Redes – Projeto e Gerenciamento de Redes Seguras.* Campus, 2000. Rio de Janeiro – RJ.

WHITAKER, Andrew. SHAW, Marianne e GRIBBLE, Steven. *Denali: A Scalable Isolation Kernel*. Proceedings of the Tenth ACM SIGOPS European Workshop, Saint-Emilion – França, 2002.