UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

WILLIAM PAGNAN

MÁQUINAS VIRTUAIS NA PROTEÇÃO DE SISTEMAS DE DETECÇÃO DE INTRUSÃO

CRICIUMA, DEZEMBRO DE 2010

WILLIAM PAGNAN

MÁQUINAS VIRTUAIS NA PROTEÇÃO DE SISTEMAS DE DETECÇÃO DE INTRUSÃO

Trabalho de Conclusão de Curso apresentado para obtenção do Grau de Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense.

.

Orientador: Prof. MSc. Paulo João Martins

CRICIUMA, DEZEMBRO DE 2010

WILLIAM PAGNAN

Máquinas Virtuais na Proteção de Sistemas de Detecção de Intrusão

Submetido ao corpo docente do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Profa. MSc. Ana Claudia Garcia Barbosa
Coordenadora do Curso de Ciência da Computação

Banca Examinadora:

Prof. MSc. Paulo João Martins (UNESC)
Orientador

Prof. MSc. Rogério Antônio Casagrande (UNESC)

Prof. Esp. Fabrício Giordani (UNESC)

Dedico este trabalho a todos os meus familiares que sabem da importância do mesmo para minha realização profissional.

AGRADECIMENTOS

Agradeço ao meu orientador Prof. MSc. Paulo João Martins, responsável pela orientação durante todo o semestre, realizado com muita eficácia, persistência e fé.

Agradeço também aos meus pais, Nelson e Lucir, que sempre me apoiaram de forma confiante.

E em especial a minha namorada Cintia que sempre acreditou em minha capacidade e me deu apoio em todos os momentos para que eu conseguisse chegar ao final.

Meus sinceros agradecimentos.

RESUMO

Atualmente existe uma grande preocupação no que se trata em segurança da informação, existem diversas ferramentas que contribuem para o aumento da segurança de um sistema computacional. Dentre essas ferramentas pode-se destacar os sistemas de detecção de intrusão, esses têm por principal funcionabilidade monitorar todo o tráfego de rede. Porém são vulneráveis, pois um ataque bem sucedido pode apagar qualquer rastro que alguém tenha deixado no computador. Este trabalho apresenta uma arquitetura para uso de detectores de intrusão através da utilização de máquinas virtuais, que a cada dia vem se disseminando mais, com as principais vantagens de portabilidade e custo. A arquitetura proposta faz uso de máquinas virtuais para deixar o sistema de detecção de intrusão encapsulado tornando-o invisível e inacessível. Os testes apresentados, mostram a importância do uso de máquinas virtuais, e a capacidade de isolamento que as mesmas permitem, podendo ser usadas para encapsular sistemas de detecção de intrusão.

Palavras-chave: Segurança da informação; Maquina Virtual; IDS; Snort.

ABSTRACT

Currently there is a big concern when it comes to information security, there are several tools that help increase the security of a computer system. Among these tools can highlight the intrusion detection systems, these have the main functionality to monitor all network traffic. But they are vulnerable, because a successful attack can erase any trace that someone has left on the computer. This paper presents an architecture for use of intrusion detection by using virtual machines, each day that has been spread out more, with the main advantages of portability and cost. The proposed architecture makes use of virtual machines to stop the intrusion detection system encapsulated making it invisible and inaccessible. The tests presented show the importance of using virtual machines, and the ability of isolation that they allow, can be used to encapsulate intrusion detection systems.

Keywords: Information Security, Virtual Machine, IDS, Snort.

LISTA DE ILUSTRAÇÕES

Figura 1. Camadas do sistema operacional.....17

Figura 2. Arquitetura x86.....22

Figura 3. Maquina virtual do tipo I.....29

Figura 4. Adeos interagindo com os sistemas operacionais.....30

Figura 5. Modelo geral de arquitetura do Persus.....31

Figura 6. Maquina virtual do tipo II.....32

Figura 7. Arquitetura Denali.....39

Figura 8. Arquitetura do Snort.....56

Figura 9. Componentes do Snort.....58

Figura 10. O Snort e o modelo TCP / IP.....59

Figura 11. Plugins de saída.....64

Figura 12. Ambiente montado para realização de testes.....70

Figura 13. Sistemas operacionais.....71

Figura 14. Alteração da placa de rede do ambiente virtual.....72

Figura 15. Tela principal do BASE.....73

Figura 16. Snort executando.....74

Figura 17. Resultado após varredura de portas.....75

Figura 18. Resultado parcial após varredura.....75

Figura 19. Leitura de serviços em execução.....76

Figura 20. Varredura detalhada 77

Figura 21. Processos encontrados em execução.....78

Figura 22. Serviços encontrados em execução.....78

LISTA DE SIGLAS

API	*Application Programming Interface*
BSD	*Berkeley Software Distribution*
CISC	*Complex Intruction Set Computing*
CVM	*Cooperative Virtual Machine*
DMA	*Direct Memory Access*
DoS	*Denial of Service*
FTP	*File Transfer Protocol*
HIDS	*Host Based Intruction Detection*
HTTP	*HyperText Transfer Protocol*
IDS	*Intrusion Detection System*
IP	*Internet Protocol*
IIS	*Internet Information Services*
ISS	*Internet Security Systems*
JVM	*Java Virtual Machine*
MMU	*Memory Management Unit*
NIDS	*Network Based Intrusion Detection*
OTN	*Options Tree Nodes*
RISC	*Reduced Intruction Set Computing*
RPC	*Remote Procedure Call*
RTN	*Rule Tree Nodes*
SCSI	*Small Computer System Interface*
SMTP	*Simple Mail Transfer Protocol*
SVM	*Security and Virtual Machine*

TCP	*Transmission Control Protocol*
TLB	*Translation Lookaside Buffer*
UCP	Unidade Central de Processamento
UDP	*User Datagram Protocol*
USB	*Universal Serial Bus*
VM	*Virtual Machine*
VMM	*Virtual Machine Monitor*
VMX	*Virtual Machine Extensions*

SUMÁRIO

1 **INTRODUÇÃO** 13

1.1 OBJETIVO GERAL 13

1.2 OBJETIVOS ESPECÍFICOS 14

1.3 JUSTIFICATIVA 14

2 **MÁQUINAS VIRTUAIS** 16

2.1 TÉCNICAS DE VIRTUALIZAÇÃO 19

2.2 SUPORTE EM HARDWARE À VIRTUALIZAÇÃO 20

2.3 ARQUITETURA x86 21

2.3.1 AMD Pacifica 23

2.3.2 Intel Vanderpool 23

2.4 USO DE MÁQUINAS VIRTUAIS 24

2.5 VANTAGENS E DESVANTAGENS DAS MÁQUINAS VIRTUAIS 26

2.6 PROJETOS DE MÁQUINAS VIRTUAIS 27

2.6.1 Emuladores 27

2.6.1.1 Bochs 28

2.6.1.2 QEMU 28

2.6.2 VMM TIPO I 29

2.6.2.1 Adeos 29

2.6.2.2 Perseus 31

2.6.2.3 Plex86 32

2.6.3 VMM TIPO II 32

2.6.3.1 Microsoft Virtual PC 33

2.6.3.2 User-Mode Linux 33

2.6.3.3 coLinux 34

2.6.3.4 VMware 35
2.6.3.5 VirtualBox 37
2.6.4 PARAVIRTUALIZAÇÃO 38
2.6.4.1 Denali 38
2.6.4.2 Xen 39
2.6.5 Virtualização de linguagens de alto nível 41
2.6.5.1 Java Virtual Machine 41
2.6.6 Virtualização no nível do sistema operacional 42
2.6.6.1 FreeBSD Jail 42
2.6.6.2 Linux-VServer 43
3 **SEGURANÇA DA INFORMAÇÃO** 45
3.1 TIPOS DE ATAQUES 46
3.1.1 Negação de serviço 47
3.1.2 Sniffers 48
3.1.3 Firewall 49
4 **IDS – INTRUSION DETECTION SYSTEM** 50
4.1 SISTEMAS DE DETECÇÃO DE INTRUSÃO BASEADOS EM REDE 51
4.2 SISTEMAS DE DETECÇÃO DE INTRUSÃO BASEADOS EM *HOST* 52
4.3 SISTEMAS DE DETECÇÃO DE INTRUSÃO HÍBRIDOS 53
5 **EXEMPLOS DE SISTEMAS DE DETECÇÃO DE INTRUSÃO** 54
5.1 SNORT 54
5.1.1 Componentes do Snort 57
5.1.2 Mecanismo de Captura 59
5.1.3 Pré-processadores 60
5.1.4 Mecanismo de Detecção 61

5.1.5 Plugins de Saída 63
5.2 REALSECURE 64
5.3 INTRUDER ALERT 65
5.4 NUZZLER BASIC 66
6 **TRABALHOS CORRELATOS** 67
6.1 PROTEÇÃO DE DETECTORES DE INTRUSÃO ATRAVÉS DE MÁQUINAS VIRTUAIS 67
6.2 UTILIZAÇÃO DE MÁQUINAS VIRTUAIS PARA IMPLANTAR UM MECANISMO TRANSPARENTE DE DETECÇÃO DE INTRUSÃO EM SERVIDORES WEB 67
6.3 DETECTANDO INTRUSÕES NA MÁQUINA VIRTUAL USER-MODE LINUX 67
7 **IMPLEMENTAÇÃO DE UM SISTEMA IDS EM UMA MÁQUINA VIRTUAL** 69
7.1 ESCOLHA DOS SOFTWARES UTILIZADOS NOS TESTES 69
7.2 ESTUDO DE CASO: MÁQUINAS VIRTUAIS NA PROTEÇÃO DE SISTEMAS DE DETECÇÃO DE INTRUSÃO 70
7.3 REALIZAÇÃO DE TESTES NA MÁQUINA VIRTUAL 74
7.4 REALIZAÇÃO DE TESTES NO SISTEMA HOSPEDEIRO 76
7.5 DIFICULDADES ENCONTRADAS 79
8 **CONCLUSÃO** 80
REFERÊNCIAS 81
APÊNDICE A – INSTALAÇÃO DO SISTEMA IDS SNORT USANDO APACHE, MYSQL E BASE 83
APÊNDICE B – ARTIGO 94

1 INTRODUÇÃO

A utilização de ambientes computacionais tem se difundido de forma crescente, disponibilizando informações importantes para o dia-a-dia das pessoas e das organizações. Porém, com a disseminação destes ambientes aumenta a preocupação no que se refere à segurança dos dados e informações, pois estas não devem ser acessadas por pessoas não autorizadas. Existem pessoas (*crackers*) que visam acessar sistemas computacionais a fim de alterar, apagar dados, entre outros, prejudicando de alguma forma as instituições.

Assim, a fim de proporcionar segurança nesses sistemas, bem como aos seus usuários, existem diferentes alternativas que objetivam proteger os dados, como por exemplo, *firewall,* virtualização de sistemas operacionais e sistemas de detecção de intrusão (*Intrusion Detection System* – IDS). É crescente o uso de sistemas virtualizados, já que os mesmos podem ser usados para ampliar o nível de segurança de um ambiente computacional contra ataques ou intrusões indesejadas (CHEN, 2001).

Além disso, pode-se usufruir dos IDS que monitoram continuamente as atividades de um ambiente computacional, visando encontrar evidências de intrusão (LAUREANO, 2004).

Este trabalho nos mostra algumas vantagens que uma arquitetura virtual nos proporciona, e como utilizá-la na segurança de ambientes de detecção de intrusão.

1.1 OBJETIVO GERAL

Implementar e demonstrar o uso de virtualização na proteção ao sistema de detecção de intrusão.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos desta pesquisa são:

a) Compreender e aplicar o processo de virtualização;
b) Entender e utilizar um sistema de detecção de intrusão;
c) Analisar e descrever algumas ferramentas de virtualização e detecção de intrusão;
d) Estudo de caso no uso de máquina virtual na proteção de sistemas de detecção de intrusão.

1.3 JUSTIFICATIVA

Atualmente na Internet existem muitos programas que visam comprometer o funcionamento de um computador que está permanentemente conectado a uma rede. Estes computadores podem se tornar alvos dos mais variados tipos de ataques, podendo assim ficar com seu funcionamento comprometido por algum tempo ou definitivamente. Dessa forma, exige-se do administrador do sistema um monitoramento constante. Para a proteção destes utiliza-se de vários recursos, dentre eles *firewalls*, que controlam o fluxo de dados que passam pela rede e de ambientes virtuais, entre outros.

A utilização de máquinas virtuais vem se tornando uma alternativa interessante para vários sistemas de computação, por suas vantagens em custos e portabilidade (BLUNDEN, 2002), pois através destas, por exemplo, executam-se diferentes sistemas operacionais sobre o mesmo *hardware* simultaneamente e simulam-se configurações e situações diferentes do mundo real.

O conceito de máquina virtual também pode ser empregado para melhorar a segurança de um sistema computacional contras ataques e intrusões indesejadas (CHEN,

2001), pois simula um ambiente semelhante ao real, para o acesso dos usuários, sendo que se ocorrer uma invasão, está será no ambiente virtual, protegendo assim a aplicação real da invasão. A virtualização também pode ser empregada junto a ferramentas de Sistema de Detecção de Intrusão (*Intrusion Detection System* - IDS). Dessa forma, a implementação de ferramentas IDS com o auxilio de máquinas virtuais, aumentará o nível de segurança no ambiente.

1.4 ESTRUTURA DO TRABALHO

Este trabalho é composto dos seguintes capítulos:

Capítulo 2: aborda o conceito de máquinas virtuais, mostrando a base de seu funcionamento e suas principais características;

Capítulo 3: mostra-nos a importância da informação nos dias atuais, e a dificuldade de mantê-la segura;

Capítulo 4: aborda o conceito de sistemas de detecção de intrusão, características, e tipos.

Capítulo 5: relata exemplos de sistemas de detecção de intrusão e suas características.

Capítulo 6: é realizada uma abordagem dos trabalhos correlatos.

Capítulo 7: Apresenta o trabalho desenvolvido e os testes realizados.

2 MÁQUINAS VIRTUAIS

O termo máquinas virtuais pode parecer algo recente, mas já vem sendo usado há muito tempo. Desde o início da computação ela vem sendo utilizada para compreender melhor o multiprocessamento, a multi-programação e o multi-acesso. Na década de 50 já havia pesquisadores na Inglaterra que desenvolviam o conceito de memória virtual, que consiste em desvincular os endereços físicos de memória daqueles endereços vistos pelas aplicações do sistema, permitindo assim que os programas executassem em qualquer lugar da memória (LAUREANO, 2006).

Mais tarde na década de 60 começou a ser desenvolvido máquinas virtuais que eram originalmente desenvolvidas para centralizar os sistemas de computador utilizados no ambiente VM/370 da IBM. Este sistema simulava uma cópia de uma máquina real e os usuários ao utilizarem a mesma, tinham a ilusão de o sistema ser único e exclusivo a eles, porém havia muitos conflitos de *hardware* e *software* ainda para serem resolvidos, tornando o desempenho da máquina muito baixo (CASTRO, 2006).

Por muitos anos estudaram-se ambientes compartilhados, obtendo-se alguns casos de sucesso, podendo assim resolver problemas como disponibilidade e segurança de sistemas.

Depois deste período, por volta de 1980 até o final da década de 90, as máquinas virtuais ficaram restritas aos supercomputadores e aos *mainframes*. O contínuo aumento do poder de processamento dos computadores que causou esta mudança, pois máquinas muito rápidas estão ao alcance de qualquer empresa, instituições e até mesmo usuários domésticos. Sendo assim, houve uma migração para *desktops*. Os computadores começaram a ter espaço físico, processamento e memória suficientes para suportar mais de um sistema operacional (CASTRO, 2006). Dessa forma, igual a um *mainframe* um PC pode suportar vários sistemas, compartilhando o mesmo *hardware*.

Uma máquina real é composta por vários componentes físicos que permitem que as operações de um sistema operacional e suas aplicações se realizem. Já uma máquina virtual ou Virtual Machine (VM) pode ser definida como uma cópia idêntica isolada de uma máquina real (CHEN, 2001). Com o enorme crescimento do poder de processamento dos computadores hoje em dia, a utilização de máquinas virtuais vem sendo uma ótima opção para sistemas da computação, pelos seus custos e portabilidade, inclusive para sistemas de segurança (LAUREANO, 2006), que é um dos objetivos deste trabalho de conclusão.

Cada máquina virtual é completamente isolada uma das outras, sendo assim, sua utilização proporciona um aumento de segurança para os recursos do sistema, pois os mesmos estão de certa forma protegidos. As aplicações não confiáveis podem ser executadas em máquinas virtuais separadas, isolando assim o perigo de danos na máquina real (CASTRO; SANTOS; GEUS, 2004).

As máquinas virtuais proporcionam um isolamento entre o sistema operacional convidado e sistema operacional hospedeiro, sendo assim um ataque ao obter sucesso na invasão da aplicação e dominar o sistema convidado, tem ainda que comprometer o *software* da máquina virtual para depois poder atingir a máquina real (CASTRO; SANTOS; GEUS, 2004).

Os sistemas operacionais e aplicações são desenvolvidos para aproveitar ao máximo dos recursos do *hardware*, porém ao longo dos anos muitas plataformas operacionais diferentes foram criadas, causando incompatibilidades entre si. O uso de máquinas virtuais permite compatibilizar diferentes plataformas, pois ela ao ser instalada na máquina real cria uma “camada” que é chamada de virtualização.

É um recurso de software que possibilita a execução virtual de um ou mais sistemas operacionais ao mesmo tempo em apenas uma máquina física, que já contém um sistema operacional nativo instalado. Primeiramente é instalado um software no sistema

operacional da máquina real, e a partir deste, outros sistemas operacionais podem ser adicionados de forma virtual (BRITO et al., 2008).

Atualmente este tem chamado muita a atenção de pequenas e grandes empresas, pois a utilização desse recurso pode trazer diversos benefícios, como economia de energia elétrica e de espaço. Sendo assim as empresas podem eliminar custos sem trazerem impactos a funcionalidades de seus equipamentos (BRITO et al., 2008).

Segundo Laureano (2006) uma máquina virtual é um ambiente criado por um monitor de máquinas virtuais ou *Virtual Machine Monitor* (VMM) também conhecido como hypervisor, que é responsável por simular os recursos de *hardware* para a utilização das máquinas virtuais e traduzir suas requisições para a máquina real. O VMM fornece uma interface muito perfeita do *hardware*, de maneira que as aplicações e sistemas que estão em cada VM "pensam" estar trabalhando em uma máquina real.

Para que possamos entender um pouco melhor o funcionamento de uma máquina virtual é necessário que se faça uma pequena análise paralela do funcionamento de um sistema operacional. A principal função de um sistema operacional é gerenciar todos os recursos que são disponibilizados as aplicações, para que isso ocorra somente o sistema operacional pode ter acesso direto aos recursos, dando as aplicações apenas acesso indireto por meio de uma interface provida pelo mesmo, chamada de Interface de Programação das Aplicações ou *Application Programming Interface* (API) (CASTRO, 2006).

Quando um sistema operacional recebe uma chamada pela API seu núcleo gerencia eventuais concorrências por recursos, e executa as operações requisitadas através de uma camada interligada ao *hardware*, que transforma os comandos de alto nível do sistema operacional em comandos de baixo nível específicos aos periféricos acionados (CASTRO; SANTOS; GEUS, 2004).

Observando uma estrutura simplificada de um sistema (Figura 1), conclui-se que com a substituição da camada dependente de *hardware* do sistema convidado por outra camada que tenha a mesma interface com núcleo, mas que utilize as chamadas da API do sistema operacional hospedeiro ou a emulação da saída da camada dependente do *hardware* do sistema convidado para uma API do sistema anfitrião torna-se possível executa-lo como uma aplicação de outro (CASTRO; SANTOS; GEUS, 2004).

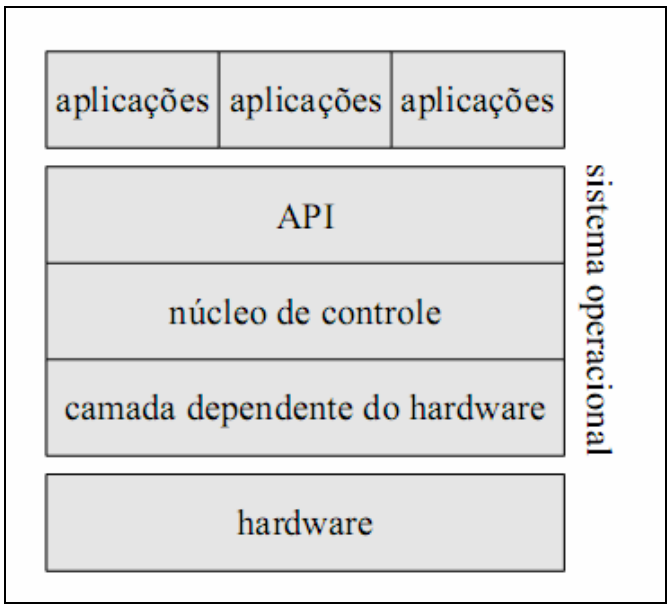

Figura 1. Camadas do sistema operacional.
Fonte: (CASTRO; SANTOS; GEUS, 2004).

Desta forma, defini-se uma máquina virtual como sendo uma abstração em *software* de um *hardware*, que pode ser executado sobre outro sistema operacional e que permita que outras aplicações sejam executadas, inclusive se estas forem incompatíveis com o sistema hospedeiro (LAUREANO, 2006).

2.1 TÉCNICAS DE VIRTUALIZAÇÃO

As técnicas mais utilizadas atualmente para a virtualização são a paravirtualização, virtualização total e recompilação dinâmica.

Na paravirtualização é necessário fazer modificações no sistema virtualizado (sistema convidado), para que o mesmo consiga acessar recursos do *hardware* diretamente. O acesso é monitorado pelo monitor de máquinas virtuais, que repassa ao sistema todas as informações necessárias para o acesso. A paravirtualização é umas das técnicas que tem a melhor *performance*, pois ela reduz a complexidade das máquinas virtuais, sendo assim compensa as modificações que terão que ser feitas nos sistemas convidados (LAUREANO, 2006).

Já na virtualização total o sistema não sofre qualquer alteração, sendo este o principal benefício desta técnica, porém esta técnica torna-o mais lento e precisa da intervenção do monitor de máquinas virtuais para o gerenciamento do acesso a memória, ao disco e para implementar alternativas para que operações privilegiadas possam ser executadas em certos processadores que não suportam a virtualização nativa (LAUREANO, 2006).

A recompilação dinâmica também é muito utilizada, essa técnica faz a compilação de partes do código durante a execução do sistema, podendo assim ajustar o código gerado, produzindo um ambiente idêntico ao original para que o programa consiga informações que não estão disponíveis em um compilador estático tradicional. Ela também pode ser aplicada como parte de uma estratégia de otimização adaptável para executar uma representação portátil do programa (LAUREANO, 2006).

2.2 SUPORTE EM HARDWARE À VIRTUALIZAÇÃO

Em execução uma máquina virtual faz com que o sistema operacional “pense” que tem acesso irrestrito ao *hardware* da máquina real, sendo que se fosse verdade, poderia causar diversos problemas. Sendo assim, o sistema da máquina virtual tem que impedir que

determinadas instruções sejam executadas, utilizando de mecanismos de proteção de acesso ao *hardware* (SILBERSCHATZ; GALVIN; GAGNE, 2001).

Os primeiros computadores permitiam que apenas um processo fosse executado, mais com a difusão do uso de computadores, os sistemas operacionais começaram a permitir que os recursos fossem divididos entre vários processos. Mais com isso surgiu um problema: se o programa que estivesse em execução falhasse, ele poderia afetar o funcionamento de outros programas ou do sistema operacional (SILBERSCHATZ; GALVIN; GAGNE, 2001).

O *hardware* detecta muito dos erros dos programas e gera uma interrupção no processamento para que o sistema operacional consiga contornar a situação, forçando o término do programa com problema. Mas para que isso aconteça, o *hardware* necessita de dois modos de operação, o modo usuário e o modo supervisor. Sendo assim, o sistema operacional é carregado em modo supervisor na hora do *boot*, e os programas são carregados em modo usuário, protegendo o sistema operacional e os programas de algum erro, separando as instruções que podem causar problemas, das "instruções privilegiadas", permitindo sua execução apenas em modo supervisor (SILBERSCHATZ; GALVIN; GAGNE, 2001).

Existem também algumas instruções chamadas de "instruções sensíveis", que podem causar inconsistência no funcionamento do sistema quando executadas dentro de uma máquina virtual, por isso a máquina tem que mascarar a execução dessas instruções.

2.3 ARQUITETURA x86

A arquitetura x86 (ou IA-32) tem quatro níveis de privilégios, esses níveis são numerados de 0 (com maior privilegio) a 3 (com menor). O nível 0 é utilizado pelo *Kernel* e por outras partes críticas do sistema operacional, o 1 e 2 por softwares menos críticos e o nível 3 fica para os programas de aplicação.

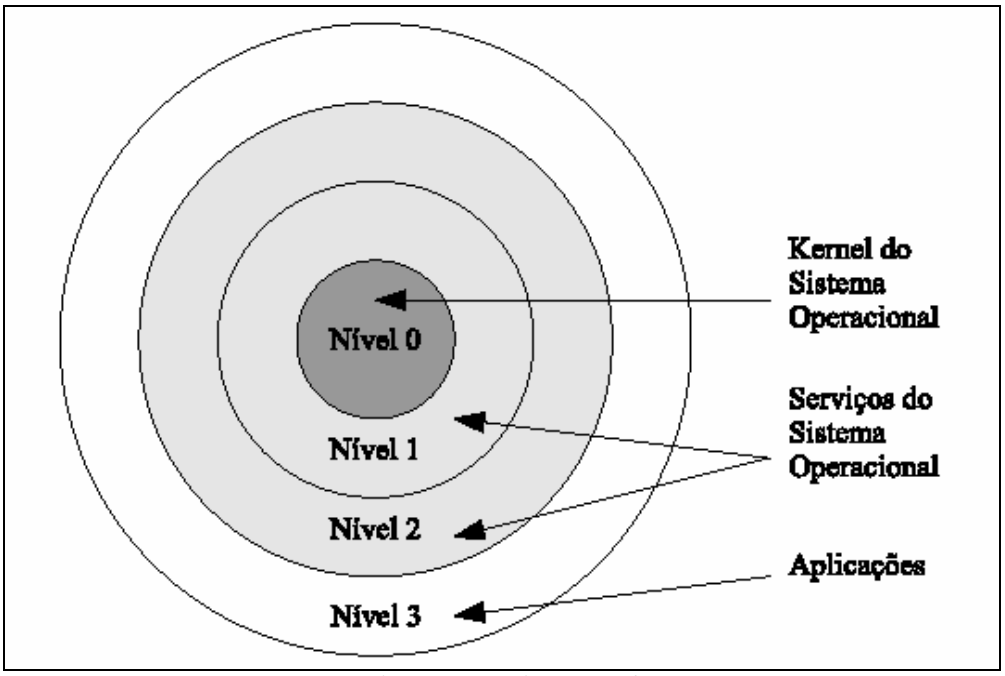

Figura 2. Arquitetura x86
Fonte: CASTRO, A. B. (2006)

A comunicação entre os níveis é feita por meio de uma interface bem controlada e protegida chamada de *gate*[1]. Qualquer acesso há um nível mais privilegiado que não seja por meio dessa interface e não tenha privilégio suficiente gera uma exceção de proteção geral.

Segundo Castro (2006) neste tipo de arquitetura existem instruções sensíveis que não são privilegiadas, isso torna a arquitetura x86 não virtualizável, entretanto isso não impede a execução de máquinas virtuais, apenas torna a tarefa mais dificultosa, pois a VM tem que rastrear a execução de todas as instruções proibidas.

A AMD e a Intel estão em constante desenvolvimento dos projetos para tornar suas arquiteturas virtualizáveis. O projeto da AMD é chamado de *Pacifica* ou *Security and Virtual Machine* (SVM), e o da Intel de *Vanderpool* ou *Virtual Machine Extensions* (VMX).

[1] Interface controladora de níveis de acesso

2.3.1 **AMD Pacifica**

A AMD com o apoio de diversas empresas que têm muito interesse na tecnologia de virtualização, incluindo Microsoft Corp., SunMicrosystems Inc., XenSource entre outras modificou o processador da linha AMD Opteron (64-bit e 32-bit) para suportar a virtualização.

Este novo processador também conhecido como SVM, é um conjunto de extensões que dá suporte em *hardware* a virtualização e a segurança. A nova arquitetura de máquinas virtuais da AMD provê as seguintes inovações (CASTRO, 2006):

a) mecanismos de troca de contexto entre o VMM e o sistema virtual;
b) habilidade de interceptar determinadas instruções do sistema virtual;
c) proteção ao acesso direto à memória - *Direct Memory Access* (DMA);
d) suporte a interrupções virtuais;
e) marcas no *Translation Lookaside Buffer* (TLB), para diminuir a sobrecarga da virtualização.

2.3.2 **Intel Vanderpool**

A Intel implantou suporte a virtualização nos processadores Intel Itanium 2 e Xeon 64-bit na linha de servidores, e já está incluíndo este suporte em *desktops* e *laptops*.

Para os processadores conhecidos como x86 (IA-32), a Intel desenvolveu uma tecnologia chamada VT-x, que nada mais é que um conjunto de extensões que permite a virtualização em *hardware*, chamado VMX. Com esse novo suporte cria-se uma nova operação, que tem dois modos: operação *root* e operação não *root*. Sendo assim, o VMM é

respectivamente executado no primeiro modo e o sistema virtual é executado no segundo (com acesso restrito e modificado) (CASTRO, 2006).

2.4 USO DE MÁQUINAS VIRTUAIS

Segundo Laureano (2006) podemos utilizar máquinas virtuais nos mais vários meios, sendo os principais descritos abaixo:

a) **ensino:** podem-se usar máquinas virtuais, por exemplo, no treinamento para administração de sistemas, pois a exigência de equipamentos dedicados a este tipo de ensino acaba tornando o treinamento muito caro (equipamentos), sem falar nos problemas que acontecem durante o curso, como reinstalação do sistema operacional e aplicações, perda de configurações, entre outros;

b) **consolidação de servidores:** consiste em centralizar e diminuir o número de equipamentos e de aplicações instaladas nos servidores da organização, com o objetivo de aumentar a segurança, reduzir a manutenção, economizar em recursos financeiros e físicos;

c) **plano de contingência**: este plano assegura a disponibilidade de recursos de sistemas críticos e facilita a continuidade das operações durante uma crise. Um servidor virtual pode ser recuperado de forma muito rápida, garantindo uma maior disponibilidade dos sistemas, pois os administradores podem fazer *backup* dos servidores de missão-crítica em máquinas virtuais. Essas máquinas são salvas em formatos de arquivos, por isso podem-se salvar várias instâncias do mesmo servidor, e em caso de falha, uma nova cópia pode ser inicializada;

d) **migração de aplicações:** em alguns casos as corporações precisam manter suas aplicações em funcionamento, mas há necessidade de substituir-se o *hardware*, porém algumas aplicações não podem ser modificadas para acompanhar essa evolução. A utilização de máquinas virtuais é uma ótima opção nesse caso, pois é possível instalar e configurar uma VM sobre um *hardware* avançado, e instalar um sistema operacional antigo.

   Podemos utilizar máquinas virtuais para migrar aplicações de plataformas diferentes, como de uma plataforma Linux para Windows;

e) **confinamento de processos**: um programa para poder executar dentro de um sistema operacional precisa de algumas permissões, ou o usuário que está executando esse programa precisa ter essas permissões para que o mesmo funcione. Porém quando se executa esse processo ele herda todas as permissões de quem o executou, daí o problema, se este processo estiver vulnerável a ataques externos, ele pode permitir ao atacante obter acessos ao sistema indesejados, podendo mudar configurações e permissões entre outras.

   As máquinas virtuais podem ser utilizadas para confinar esses processos, pois elas possuem uma propriedade que garante a segurança, que é o isolamento de processos;

f) **honeypots**: é um sistema colocado em rede para atrair invasores e ser comprometido e depois analisado posteriormente. Uma honeynets é composta de inúmeros honeypots que são ferramentas com a qual os invasores interagem, o que pode ser perigoso se mal implementada, por isso deve ser utilizado junto a outras ferramentas de proteção. A equipe da Honeynet Project já conseguiu implementar a criação de um honeynet com o uso de máquinas virtuais. O processo de criação é idêntico, porém em vez de se ter

diversos equipamentos distintos para realizar esta função, em apenas um equipamento são executadas várias instâncias virtuais para todas as funções;

g) **detecção de intrusão**: são sistemas que tem por principal objetivo detectar se há princípios de ataques no sistema, ou se algum usuário está fazendo mau uso do sistema. Estes tipos de sistemas usam técnicas de inspeção na rede, todas objetivam de alguma forma monitorar a rede, buscando anomalias ou irregularidades.

2.5 VANTAGENS E DESVANTAGENS DAS MÁQUINAS VIRTUAIS

As máquinas virtuais trazem diversas vantagens para os sistemas de computação, sendo as principais (RAITZ, 2005):

a) executar diferentes sistemas operacionais sobre o mesmo *hardware*, ao mesmo tempo;
b) facilitar testes de novos sistemas operacionais;
c) simular configurações e situações diferentes do mundo real;
d) garantir a portabilidade de aplicações;
e) diminuição de custos de *hardware*;
f) facilidade na migração, gerenciamento e replicação de aplicações ou sistemas operacionais;
g) simular alterações e falhas no *hardware* para testes ou reconfigurações de sistemas operacionais, aumentando a confiabilidade e escalabilidade dessas aplicações.

As máquinas virtuais também trazem com si algumas dificuldades, sendo as principais encontradas as seguintes (LAUREANO, 2006):

a) custo do processo de virtualização;
b) processador não virtualizado;
c) diversidade de equipamentos, sendo assim, o trabalho do monitor de máquinas virtuais é sobrecarregado;
d) preexistência de softwares já instalados nos PCs ou desktops, onde a máquina virtual tem que ser disponibilizada, mas estas não podem afetar os sistemas já existentes na máquina;
e) custo adicional de execução dos processos em comparação á de uma máquina real.

2.6 PROJETOS DE MÁQUINAS VIRTUAIS

O objetivo dessa sessão é apresentar alguns projetos de máquinas virtuais existentes, para isso é necessário que seja dado uma visão também de suas categorias. Podemos ressaltar as seguintes (CASTRO, 2006): emuladores; VMM do tipo I; VMM do tipo II; paravirtualização; virtualização de linguagem de alto nível e no nível do sistema operacional.

2.6.1 **Emuladores**

Emuladores são softwares que tem por função principal transcreverem instruções de um processador alvo para um processador no qual estão sendo rodados. Permitem que programas ou sistemas que só executariam em uma determinada plataforma sejam executados em outras. Em um sistema emulado dificilmente se atingirá a velocidade de um sistema

original, pois o mesmo exige grande poder de processamento e memória da máquina. Exemplo: Bochs, QEMU, entre outros.

2.6.1.1 **Bochs**

É um emulador que simula totalmente a arquitetura x86 de código livre e altamente portátil. Ele interpreta todas as instruções desde o boot do sistema operacional, ele possui um ótimo suporte aos mais variados tipos de dispositivos e periféricos, tornando seu uso com qualquer sistema convidado (SANTOS, 2005).

Uma das grandes desvantagens desse sistema é a utilização da simulação em *software* para cada instrução da plataforma x86, trazendo assim um desempenho baixo neste tipo de máquina. Mais por outro lado, os sistemas executados por esse emulador não precisam de modificações. Ele emula o 386, 486, Pentium, Pentium Pro ou AMD64 podendo incluir instruções MMX, SSE, SSE2 e 3DNOW! (CASTRO, 2006).

2.6.1.2 **QEMU**

É um emulador genérico que usa tradução dinâmica para diminuir a sobrecarga causada pela emulação, ou seja, ele converte partes do código para que o processador execute o conjunto de instruções. Este pode trabalhar em dois modos (SANTOS, 2005):

a) **emulação total do sistema ->** é emulado o sistema completo, incluindo processador e periféricos. Neste modo podem ser rodados diferentes sistemas operacionais.

b) **emulação no modo usuário ->** este modo está disponível só para sistemas Linux. Com esse modo pode-se executar processos tanto para uma plataforma

ou outra, sendo assim, programas compilados para executar em um processador x86 pode ser executadas em PowerPC.

O QEMU não necessita de alterações no sistema anfitrião, isso facilita a sua utilização. Este emulador utiliza um tratamento preciso para exceções dos processos do sistema convidado, durante a emulação de um sistema por completo, o QEMU implementa um programa chamado de *Memory Management Unit* (MMU) para conseguir o máximo de portabilidade (LAUREANO, 2006).

2.6.2 **VMM TIPO I**

São implementados diretamente sobre o *hardware* da máquina real. O monitor tem o controle do *hardware* e cria um ambiente de máquinas virtuais, permitindo assim que cada uma, execute seu próprio sistema operacional agindo de forma semelhante a uma máquina física completa (LAUREANO, 2006), assim como mostra a Figura 3:

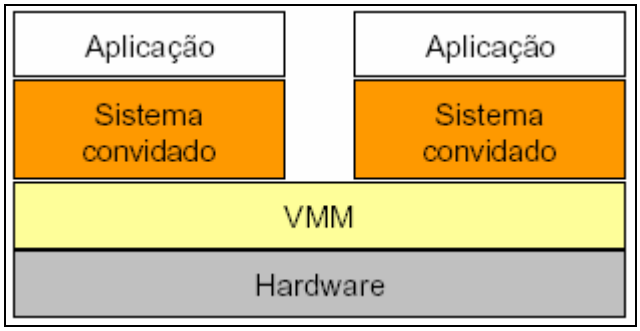

Figura 3. Máquina Virtual do Tipo I
Fonte: LAUREANO, M. A. P.; MAZIERO, C. A.; JAMHOUR, E. (2003)

As máquinas aqui abordadas como exemplos serão: Adeos, Perseus e Plex86.

2.6.2.1 **Adeos**

Os sistemas operacionais são desenvolvidos tendo controle sobre a máquina na qual estão sendo executados, sendo que os programadores fazem uso de API para ter acesso

aos recursos da máquina, e isso muitas vezes é desvantajoso para os usuários, pois eles ficam presos as API que foram desenvolvidas para seus sistemas.

Muitas vezes as funcionalidades de um sistema poderiam ser úteis para usuários de outro sistema, além dos programadores ficarem limitados aos acessos que o sistema provê a eles.

O objetivo deste emulador é disponibilizar um ambiente flexível de compartilhamento dos recursos do *hardware* entre vários sistemas operacionais, sendo que o mesmo tem que obter um estado consistente e confiável entre os sistemas. O Adeos não impõe nenhuma restrição de acesso sobre o *hardware*, mesmo que isso possa trazer algum tipo de erro no gerenciamento (CASTRO, 2006).

O Adeos cria diferentes domínios, cada um forma um ambiente, onde o sistema tem total controle. Se o sistema descobri-lo não há problema, pois o uso dos recursos não precisa ser exclusivo, o sistema pode interagir com o mesmo e permitir o acesso entre domínios diferentes, o que é mostrado na figura 4:

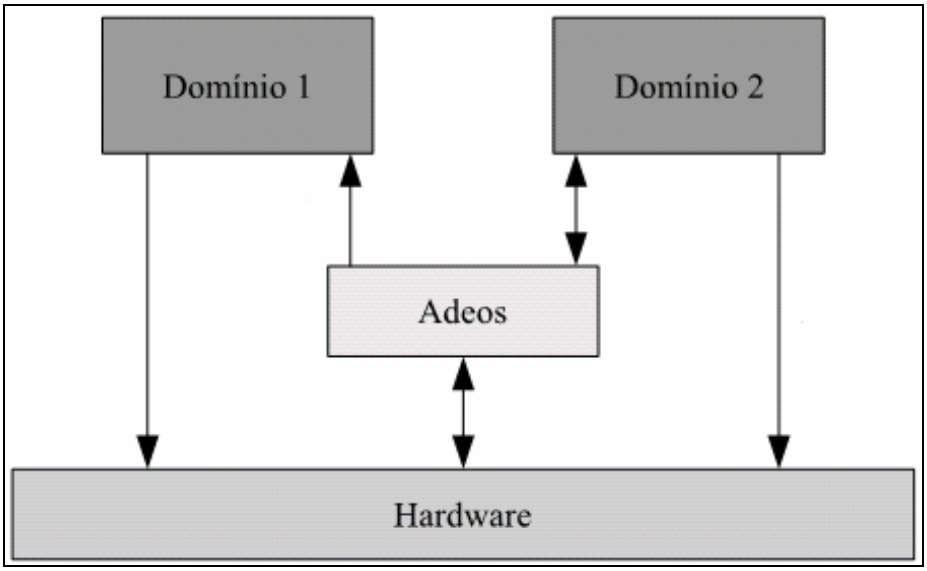

Figura 4. Adeos interagindo com os sistemas operacionais
Fonte: CASTRO, A. B. (2006)

2.6.2.2 **Perseus**

É um projeto que abrange mais a segurança, com uma arquitetura que combina projetos de sistemas operacionais modernos e tecnologias de segurança. Sendo um sistema baseado em um *microkernel*, com uma plataforma composta por vários subsistemas que possuem uma interface em que diversos sistemas operacionais podem ser instalados (CASTRO, 2006).

Na Figura 5 é demonstrada uma idéia do funcionamento geral dos componentes do Perseus, a linha vermelha divide as aplicações não confiáveis da plataforma segura do sistema. Abaixo desta linha fica a parte crítica do Perseus, que é responsável pela segurança do sistema e por isso deve ser bem protegida, só os subsistemas da parte crítica têm permissão para acessar o *hardware* diretamente.

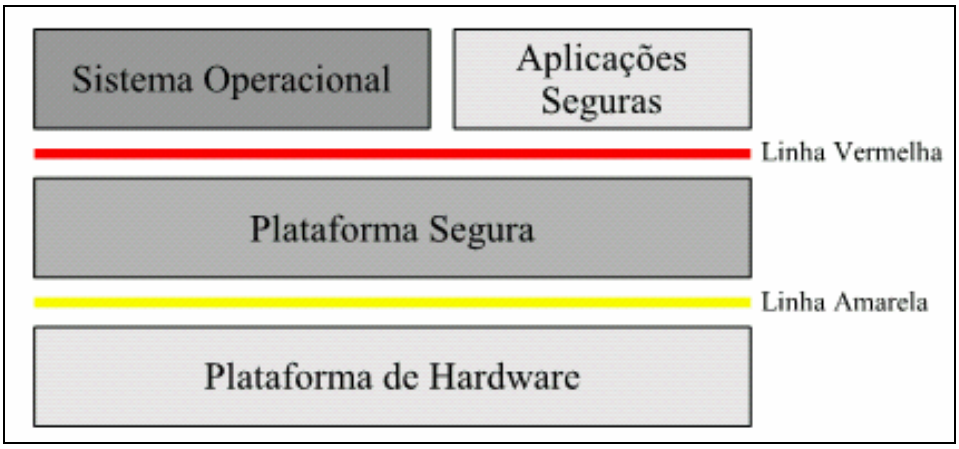

Figura 5. Modelo geral de arquitetura do Perseus
Fonte: CASTRO, A. B. (2006)

De modo geral, o sistema é divido em três camadas: o *hardware*, um gerenciador de recursos e uma camada de *software* confiável. A camada de gerenciamento de recursos tem uma interface que interage com o *hardware*, interrupções e memória. A camada de *software* é responsável por garantir a segurança dos recursos e o isolamento das aplicações, para isso ela implementa um controle de acesso.

2.6.2.3 **Plex86**

Tem por objetivo construir uma máquina virtual para Linux/x86 que tenha por principal característica ser leve. Por não implementar suporte a muitos sistemas operacionais, a máquina tende a ter um bom desempenho. Este sistema tem em seu funcionamento como base um monitor de máquinas virtuais, sendo que o mesmo requer o mínimo de modificações no Linux para ser executado. O mesmo emula apenas algumas características dos componentes principais de um sistema operacional, conseguindo assim um ganho de performance para o sistema virtualizado (CASTRO, 2006).

2.6.3 **VMM TIPO II**

Este modelo funciona como uma aplicação de um sistema operacional, ele utiliza do gerenciamento do sistema e prove mecanismos para a virtualização de um ou mais sistemas, assim como mostra a Figura 6.

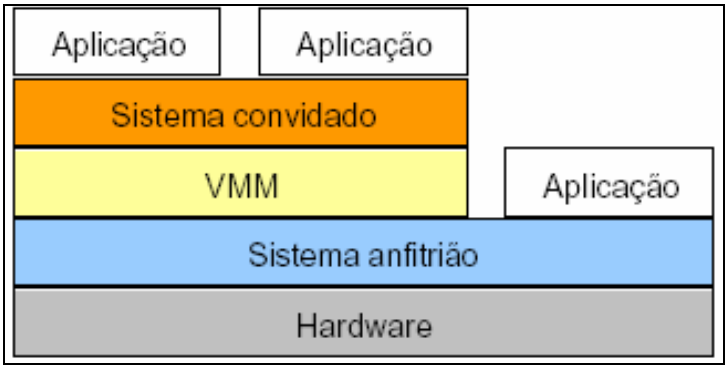

Figura 6. Máquina virtual do tipo II
Fonte: LAUREANO, M. A. P.; MAZIERO, C. A.; JAMHOUR, E. (2003)

Neste capítulo serão apresentadas algumas máquinas virtuais que utilizam deste tipo de monitor, sendo essas: Microsoft Virtual PC, User-mode Linux, coLinux, VMware e VirtualBox.

2.6.3.1 **Microsoft Virtual PC**

Ele foi originalmente projetado e comercializado pela empresa Connectix, com a idéia de dar suporte para os usuários de *Macintosh* para que os mesmos pudessem utilizar programas feitos em Windows. Seis anos mais tarde a Microsoft comprou a Connectix e deu um novo nome a esta máquina virtual de Microsoft Virtual PC (SANTOS, 2005).

Após a união das duas empresas a Microsoft comprou três sistemas: o Virtual PC, o MAC e Virtual Server. Sendo que o primeiro permite que se executem outros sistemas operacionais sobre uma máquina com Windows, o segundo permite a emulação completa de um PC para que os usuários do sistema Mac possam usufruir de outras aplicações ou sistemas. Já o Virtual Server é voltado para sistemas servidores Windows (LAUREANO, 2006).

Este sistema é de código fechado, por esse motivo poucas informações são encontradas sobre seu funcionamento, o fabricante disponibiliza apenas documentação sobre seu funcionamento e suas aplicações.

2.6.3.2 **User-Mode Linux**

Este é uma máquina virtual que executa sobre um sistema anfitrião na forma de um processo, e tudo o que é executado nela não têm acesso aos recursos do sistema anfitrião de forma direta. O monitor é o único processo que controla todas as máquinas virtuais (CASTRO, 2006).

Uma dificuldade encontrada no desenvolvimento do User-Mode Linux foi a de encontrar maneiras para virtualizar todas as capacidades do *hardware* para as chamadas de sistema do Linux, sendo a parte mais importante a distinção entre o modo privilegiado do Kernel e não-privilegiado.

O User-Mode Linux deve possuir uma distinção de privilégios equivalente, para que seu Kernel consiga acessar as chamadas do sistema anfitrião e impeça que seus próprios processos acessem diretamente os recursos subjacentes. Isso só é possível porque foi implementada na distinção de privilégios um mecanismo de interceptação de chamadas do Linux fornecida pela chamada ptrace (é uma chamada de sistema que permite observar e controlar a execução de outros processos) (LAUREANO, 2006).

O monitor ao utilizar a ptrace ganha o controle de todas as chamadas de sistema de entrada/saída geradas pelas máquinas virtuais e também todos os sinais gerados ou enviados às máquinas virtuais.

O User-Mode Linux utiliza o sistema anfitrião para operações de entrada/saída, porém para que isso ocorra com melhor desempenho é necessário alterações no sistema convidado. A virtualização de chamadas de sistema é implementada por uma *thread* de rastreamento que intercepta as chamadas do sistema e as redireciona para o *Kernel* virtual (LAUREANO, 2006).

2.6.3.3 **coLinux**

O Coopertive Linux (coLinux) é uma máquina virtual de código livre que permite executar o Linux no Windows. Esta máquina ao contrário de muitas não utiliza níveis de privilégio diferentes, ela usa a Máquina Virtual Cooperativa ou *Cooperative Virtual Machine* (CVM). A máquina sendo executada com o mesmo nível de restrição ao *hardware* fica muito mais rápida e fácil de ser implementada, porém com isso qualquer erro que ocorrer com a máquina virtual pode atingir os outros sistemas operacionais e vice-versa (LAUREANO, 2006).

Os sistemas operacionais têm seus *Kernels* transformados para trabalharem de forma cooperativamente, ou seja, eles trabalham como um time, revezando a execução. Um dos sistemas operacionais fica responsável pela alocação de memória e pelo *hardware* físico, provendo assim uma abstração (ou interface) aos outros sistemas (CASTRO, 2006).

2.6.3.4 **VMware**

É a máquina virtual muito utilizada nos dias de hoje para plataforma x86, pois o mesmo prove uma interface completa da arquitetura x86 ao sistema convidado (LAUREANO, 2006).

Mas para conseguir implementar essa interface genérica precisa de um monitor mais complexo, pois podem haver mais de um sistema operacional executando ao mesmo tempo, e para que todos os sistemas trabalhem corretamente é necessário emular algumas instruções para representar corretamente os processadores virtuais. Essas instruções são chamadas de instruções sensíveis (LAUREANO, 2006).

Para um melhor desempenho as máquinas virtuais utilizam o mecanismo trap (armadilha) do próprio processador para executar as instruções sensíveis, no entanto os processadores x86 não conseguem capturar todas essas instruções sensíveis, por isso é necessário um trabalho adicional (CASTRO, 2006).

Para conseguir capturar essas instruções sensíveis o VMware usa uma técnica chamada rescrita binária (*binary rewriting*), que analisa todas as instruções antes de serem executadas. O monitor insere um ponto de parada no lugar das instruções sensíveis, para quando executados esses pontos permitam ao processador capturar as instruções do mesmo. Essa técnica torna o monitor mais complexo, porém possibilita a implementação de um conjunto completo de instruções x86 (LAUREANO, 2006).

Para obter um melhor desempenho o monitor usa uma abordagem hibrida para implementar a interface com as máquinas virtuais. O gerenciamento da memória é realizado pela manipulação direta do *hardware* e para tornar o monitor mais simples o controle da E/S é do sistema anfitrião. No entanto, essa simplificação ocasionou uma perda de desempenho nas primeiras versões do VMware e que foram corrigidas ao longo do tempo, buscando melhorar o desempenho.

O VMware aloca uma parte da memória exclusiva para seu uso, sendo essa previamente alocada para ser usada pelo sistema convidado, evitando assim que os sistemas se colidam. Para controlar o sistema virtualizado é implementado serviços de interrupções para todas as requisições do sistema convidado, quando uma exceção acontece, ela é analisada primeiro pelo monitor depois é repassada. Já quando se trata de interrupções de E/S, elas são repassadas para o sistema anfitrião, para serem analisadas e controladas corretamente (CASTRO, 2006). Atualmente estão disponíveis três versões do VMware (LAUREANO, 2006):

a) **VMware Workstation e VMware Player** -> para aplicações mais leves ou testes, indicada para *desktops* pois permite a execução de vários sistemas operacionais no mesmo computador. Esta permite sistemas Windows e Linux, sem nenhuma alteração especial nos sistemas convidados. A versão Player é utilizada no uso de máquinas virtuais criadas e configuradas pelo VMware Workstation.

b) **VMware Server GSX** -> para aplicações profissionais em pequena escala, disponível para Linux e Windows. Esta máquina virtual tem a inserção de devices drivers[2] específicos para comunicação direta com o *hardware*, buscando um maior desempenho do sistema.

[2] *Device driver* permite que os softwares de alto nível interajam com um dispositivo de *hardware.*

c) **VMware Server ESX** -> esta versão foi criada para aplicações profissionais em larga escala ou missão crítica. O VMware Server ESX se diferencia principalmente das outras versões pelo motivo de ser uma máquina virtual do tipo I.

2.6.3.5 **VirtualBox**

Inicialmente criado pela empresa Innotek, oferecia uma licença proprietária e uma versão do produto para uso pessoal ou de avaliação sem custo. Em Janeiro de 2007 é lançado a versão VirtualBox OSE (*Open Source Edition* - OSE) com a licença GPL (*General Public License* -GPL). Em Fevereiro de 2008 a Innoteck é adquirida pela Sun Microsystems, e logo mais no dia 20 de Abril de 2009 a Oracle compra a Sun Microsystems e todos o seu produtos.

O VirtualBox é um ótimo virtualizador para *hardwares* de arquitetura x86 , tem um desenho extremamente modular com interfaces de programação interna bem definidas e um desenho cliente/servidor. Isso torna fácil o controle de várias interfaces de uma só vez.(referencia). As definições de configuração de máquinas virtuais são armazenadas em XML e são totalmente independentes das máquinas locais, ou seja, podem ser facilmente transferidas para outros computadores (MIRANDA, 2010).

Para facilitar a troca de dados entre os hospedeiros e convidados, o este *software* permite a declaração dos diretórios como "pastas compartilhadas", que pode ser acessadas de dentro do ambiente virtual. O mesmo possui uma série de recursos disponíveis, porém estes recursos só podem se usados na versão completa, um exemplo disso é o controlador virtual USB que permite arbitrariamente ligar dispositivos USB em suas máquinas virtuais sem ter que instalar um *driver* de dispositivo específico ao *host*.

Umas das grandes diferenças do VirtualBox é o fato dele apoiar-se no padrão RDP (*Remote Desktop Protocol)*, ou seja, pode atuar como um servidor, o que lhe permite "executar" a máquina virtual remotamente em alguns serviços que exibem os dados RDP (MIRANDA, 2010).

2.6.4 **PARAVIRTUALIZAÇÃO**

Esta é uma técnica onde o sistema a ser virtualizado sofre alterações para que a interação com o monitor de máquinas virtuais tenha uma melhor performance. Embora sejam necessárias mudanças no sistema convidado, o que diminui a portabilidade do mesmo, a paravirtualização permite o acesso direto ao *hardware* por parte desse sistema, este acesso é monitorado pelo monitor de máquinas virtuais, que informa os “limites” ao sistema convidado (LAUREANO, 2006).

Segundo Laureano (2006) o aumento de desempenho obtido, é a principal razão para utilizar a paravirtualização, já que tem-se que fazer modificações nos sistemas convidados. As máquinas Xen e Denali serão apresentadas nesta seção como exemplos.

2.6.4.1 **Denali**

É um projeto de máquina virtual que propõe implementar vários domínios protegidos, permitindo assim que vários servidores possam ser executados. Esses domínios devem ser: isolados entre si, suportar vários domínios, trocar rapidamente entre eles, respondendo de forma rápida aos serviços requisitados. Para que isso ocorra, há uma camada de software (figura 6) que virtualiza o *hardware* realizando um trabalho parecido como o de

um VMM, porém essa camada não é emulada ela é ligeiramente diferencia o *hardware*, buscando simplicidade, escalabilidade e desempenho (CASTRO, 2006).

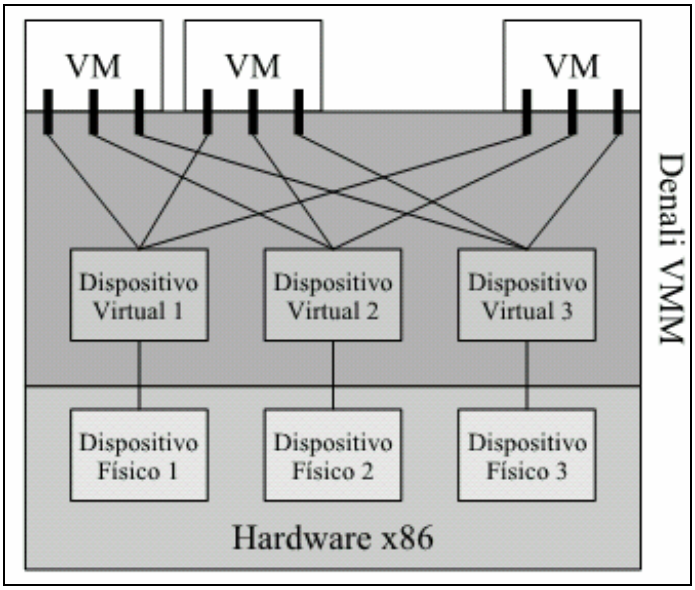

Figura 7. Arquitetura Denali
Fonte: CASTRO, A. B. (2006)

Esta máquina virtual tem uma compatibilidade baixa com os sistemas operacionais atuais, pois tem um número reduzido de dispositivos e todos realizam a maioria dos serviços com uma única instrução. Para fazer um melhor uso dessa arquitetura provida pelo Denali foi desenvolvido um sistema hospedeiro chamado *Ilwaco*, que é dotado por um porte da pilha TCP/IP do BSD, suporte a threads e a um conjunto da API POSIX.

2.6.4.2 **Xen**

É uma máquina virtual do tipo I que exige poucas modificações do sistema operacional, sendo possível executar várias instâncias do sistema de maneira isolada e com uma sobrecarga pequena causada pela virtualização. Ele baseia-se em apresentar uma abstração ligeiramente diferente do *hardware* da máquina, trazendo assim uma maior facilidade na implementação. O VMM do Xen é denominado *hypervisor*, sua execução acontece em um nível de privilégio maior que os demais sistemas. As instruções privilegiadas

são paravirtualizadas, validadas e executadas, para otimizar a execução as chamadas do sistema. São primeiramente registradas em um tratador rápido de exceções, sendo validadas só quando forem registradas na tabela de exceções do *hardware* (CASTRO, 2006).

O Xen tem acesso privilegiado ao *hardware* da máquina e os sistemas convidados utilizam deste acesso para acessar o *hardware*, a memória é separada em blocos VMM e os sistemas convidados utilizam desses blocos como quiser, tornando assim o acesso mais direto e rápido. O acesso a dispositivos como discos rígidos decorre da mesma forma, proporcionando um acesso rápido.

Utilizando a técnica de paravirtualização já proposta pelo projeto Denali anteriormente, o monitor de máquinas virtuais do Xen foi alterado principalmente nos seguintes aspectos (LAUREANO, 2006):

- Gerenciamento de memória
    a) **segmentação** -> não é possível instalar descritores de segmentação com privilégio total;
    b) **paginação** -> o sistema convidado pode ler as tabelas de páginas de memória, mas se precisar modificá-las terá que pedir validação para o monitor.
- Gerenciamento da CPU
    a) **proteção** -> o sistema convidado deve executar em um nível menos privilegiado que o monitor.
    b) **exceções** -> o sistema convidado registra uma tabela de *handlers* para controle das exceções no monitor**.**
    c) **chamadas de sistema (system calls)** -> o sistema convidado pode instalar um controlador para as chamadas de sistema, permitindo chamadas diretas de uma aplicação para o kernel do sistema convidado.

d) **interrupções ->** não há mais interrupções de *hardware*, agora um controle de eventos mais leve se encarrega disso.

e) **controle de tempo** -> ao sistema convidado é disponibilizado uma interface onde o mesmo tem acesso ao tempo "real" e "virtual".

As versões mais novas do Xen terão suporte à arquitetura 64-bit, aos processadores da Intel e AMD. A empresa Intel tem contribuído com modificações ao kernel do Xen para facilitar o suporte futuramente da arquitetura Vanderpool.

2.6.5 **Virtualização de linguagens de alto nível**

Este tipo de máquinas virtual pode ser entendido como um emulador, mais a diferença é que o *hardware* emulado não existe. Será tratada como exemplo a Máquina Virtual Java (*Java Virtual Machine*), devido a sua grande importância nos dias de hoje.

2.6.5.1 **Java Virtual Machine**

Nos dias de hoje é comum a implementação de linguagens de programação usando uma máquina virtual, um bom exemplo disso é a *Java Virtual Machine* (JVM) que a princípio foi desenvolvida para pequenas aplicações e programas de aparelhos eletroeletrônicos e que se mostrou ideal para uso na Internet. O que tornou a linguagem Java tão interessante foi que ela pode ser executada virtualmente em qualquer plataforma (LAUREANO, 2006).

É uma máquina abstrata, sendo possível executar qualquer aplicação Java sobre ela independente do *hardware* e do sistema operacional. A JVM é um conjunto de instruções que manipula áreas de memória quando está executando algum código (CASTRO, 2006).

Um programa em Java é traduzido pelo compilador e transformado para código de máquina, após isso a JVM pega esse arquivo que contém um conjunto de instruções e tabelas de símbolos, verifica a integridade e executa o código. A vantagem desse tipo de máquina é muito grande, pois garante uma maior portabilidade para todos os programas Java em códigos-fonte e compilados (LAUREANO, 2006).

2.6.6 **Virtualização no nível do sistema operacional**

Este tipo de virtualização exporta um sistema operacional como abstração de um sistema específico. Exemplos desse tipo de sistema são: FreeBSD *Jail* e Linux-VServer.

2.6.6.1 **FreeBSD Jail**

Este sistema foi desenvolvido na iniciativa de aumentar a segurança dos processos do Unix. O FreeBSD é enclausurado em um subconjunto chamado *Jail* (jaula), que aumenta a granularidade dos mecanismos de segurança. Os processos dentro da jaula têm acesso pleno aos arquivos, processos e serviços de rede que lhe são associados, não podendo acessar nada fora desse subconjunto (CASTRO, 2006).

Após um processo ser colocado nesse sistema, ele e seus descendentes estão confinados a esse ambiente e com algumas restrições, por exemplo: só é permitido a escuta de uma porta de endereço IP. O acesso a recursos e manipulação de processos é restrito e só é permitido interagir com processos do mesmo *jail*.

Algumas funcionalidades são proibidas dentro do *jail* (LAUREANO, 2006):

a) modificar módulos do *kernel* ou acessá-los;

b) alterar configurações de rede, interfaces, endereços ou tabelas de roteamento;

c) criar novos devices;

d) montar e desmontar sistemas de arquivos;

e) acesso a rwa, divert ou routing sockets;

f) acessar outros recursos de rede que não pertençam ao *jail*.

E algumas têm acesso limitado:

a) modificar o dono e os modos de acesso de qualquer arquivo, somente se for permitido;

b) apagar qualquer arquivo, só se for permitido;

c) escutar em portas reservadas TCP e UDP, somente se o endereço for da própria *jail*.

O sistema FreeBSD juntamente com o sistema *jail* implementam uma ótima opção para a execução de processos sensíveis ou que devem ficar expostos, sendo uma boa opções de aplicação de segurança.

2.6.6.2 **Linux-VServer**

Este sistema foi criado com a iniciativa de um melhor aproveitamento de processamento dos servidores Linux, já que os mesmos na maioria das vezes trabalham a baixo de suas capacidades totais. O LinuxVServer separa o espaço de usuários em unidades distintas, que são conhecidas por *Virtual Private Servers* (VPS), eles tem que ser o mais próximo possível de um servidor real para os processos que interagem dentro deles (CASTRO,2006).

No VServer não é implementado o conceito de um *kernel* para cada contexto, sendo assim o *overhead* que é causado pela virtualização, não acontece. Com a inclusão de

algumas alterações no *kernel* do sistema Linux, novas chamadas de sistema são implementadas (LAUREANO, 2006):

a) **new_s_context** -> cria um processo de contexto;

b) **set_ipv4rrot** -> possibilita ao processo de contexto obter informações sobre a rede, criando uma interface para o processo.

O VServer isola o sistema Linux em cinco grandes áreas (LAUREANO, 2006):

a) **sistema de arquivos** -> o VServer cria e instancia um sistema de arquivos e um subdiretório;

b) **processos** -> todos os processo são presos dentro de um contexto, e não enxergam os outros contextos nem o sistema real;

c) **rede** -> tem suas próprias configurações de rede**;**

d) **capacidades de superusuário (root)** -> o super usuário dentro do contexto tem um nível de privilégio inferior ao usuário *root* da máquina real;

e) **sistema V para comunicação entre processos** -> este sistema é adaptado para ser utilizado apenas dentro do contexto.

Quando o sistema Linux-VServer foi desenvolvido, foi projeto para facilitar a administração de serviços de hospedagens, mas também é utilizado para testes de novas aplicações.

3 SEGURANÇA DA INFORMAÇÃO

Podemos definir segurança da informação como sendo a tentativa de minimizar a vulnerabilidade de bens (qualquer coisa de valor) e recursos, sendo esta qualquer fraqueza que possa ser explorada para atacar o sistema (FERREIRA, 2003).

Nos dias atuais é indispensável que se adote uma política de segurança em qualquer sistema, sendo de uma organização ou particular. De maneira geral os sistemas de computação devem ter duas características: confiabilidade e disponibilidade (FERREIRA, 2003).

A confiabilidade de um sistema pode ser definida como sendo a capacidade que um sistema tem em responder a uma especificação dentro de condições definidas e durante um determinado tempo de funcionamento. Ou seja, indiferentemente do tipo de falha, seja por problemas físicos de *hardware* ou por ações mal-intencionadas de um atacante, o sistema deve continuar funcionando de acordo com a sua especificação. E a disponibilidade é fato de que o sistema deve estar sempre funcionando, estando assim sempre disponível (CAMPELLO; WEBER, 2001).

Para completar o contexto de segurança, um sistema ainda tem que dispor de integridade, autenticidade e privacidade de seus dados e informações. Muitas vezes pela falta de aplicações nessa área, podem ocorrer diversos tipos de ataques ao sistema, geralmente por *hacker* ou *cracker*.

Segundo FERREIRA (2003) um *hacker* é uma pessoa interessada nos trabalhos diversos de qualquer sistema operacional de um computador. Normalmente são programadores que têm um conhecimento avançado de sistemas operacionais e linguagens de programação, e por isso podem descobrir brechas dentro de sistemas. Estão sempre em busca

de novos conhecimentos, compartilhando livremente o que eles descobrem, mas jamais corrompem dados intencionalmente.

Entretanto um *cracker*, é alguém que viola a integridade de um sistema de máquinas remotas com a intenção maliciosa. Eles destroem dados vitais, negam serviços de usuários legítimos ou simplesmente trazem problemas para seus alvos (ANÔNIMO, 2000).

De uma forma geral podemos dizer que os *hackers* descobrem novas brechas e novos tipos de ataques com a intenção de corrigi-los, já os *crackers* usam estes ataques ou brechas para invadir.

3.1 TIPOS DE ATAQUES

Podemos classificar uma ameaça como sendo a possibilidade de violação da segurança de um sistema. Logo abaixo algumas ameaças sofridas pelos sistemas computacionais (FERREIRA, 2003):

a) destruição de informação ou de outros recursos;

b) modificação ou deturpação da informação;

c) roubo, remoção ou perda de informações ou de recursos;

d) revelação de informações;

e) interrupção de serviços.

Algumas destas ameaças juntamente com uma ação intencional, podem originar um ataque. Esses podem variar, sendo considerados acidentais, maliciosos, internos ou externos.

Com a falta de segurança presente na implementação do protocolo TCP/IP, existem basicamente duas formas de um ataque acontecer (FERREIRA, 2003):

a) **ataque passivo** -> consiste no monitoramento de toda a transmissão de dados da rede com a intenção de fazer um cópia dos mesmos;

b) **ataque ativo** -> o atacante intercepta e altera os dados transmitidos.

As etapas mais comuns são: escolha do alvo, levantamento das informações, teste de ferramentas, aplicação das ferramentas contra o alvo e exploração do resultado do ataque (ANÔNIMO, 2000). Logo abaixo veremos alguns exemplos.

3.1.1 **Negação de serviço**

O *Denial of Service* (DoS) é um tipo de ataque em que há uma perda de serviço ou um impedimento para executá-lo. Isso pode durar minutos, horas ou dias, prejudicando ou mesmo derrubando a operação do sistema (FERREIRA, 2003).

A negação de serviço pode ocorrer de três maneiras: consumo de largura de banda, saturação de recursos e queda de sistema e aplicativo. No primeiro caso é um ataque em que um ou mais computadores invasores utilizam o total de banda de uma rede, o que torna sua resposta lenta ou para totalmente o servidor. Exemplo comum desse tipo, inclui o envio de dados em massa a um roteador, fazendo com que o mesmo falhe sem poder processar todo o tráfego enviado.

Outro tipo de ataque é a saturação de recursos, que consiste em ocasionar uma lentidão em diversos sistemas que fornecem serviços como correio eletrônico, protocolo de transferência de arquivos (*File Transfer Protocol* – FTP), entre outros. As limitações de recursos que acontecem com uma rede também ocorrem num sistema de computador na parte de memória, capacidade de processamento e armazenamento.

A queda de sistema e aplicativo trata-se de mais uma forma de negação de serviço, na qual uma deficiência na programação é explorada e causa a queda do sistema

operacional ou do aplicativo que está sendo executado. Muitos desses ataques são gerados contra dispositivos de rede como roteadores, *switches* gerenciados entre outros. Esses dispositivos geralmente possuem uma interface de gerenciamento que é prejudicado por meio de vários processos inadequados como estouro de *buffer*, por exemplo (FERREIRA, 2003).

Existem também, os ataques de negação de serviço locais, quando um usuário que tem acesso local à máquina da vítima pode saturar os seus recursos. Alguns sistemas Linux já possuem muitas formas de se proteger destes ataques locais, como por exemplo, o uso de limites de memória e processos abertos por usuário.

3.1.2 **Sniffers**

São dispositivos que capturam pacotes na rede, sendo que o propósito legítimo dos *sniffers*, é analisar tráfego e identificar áreas potenciais de preocupação. Eles podem variar de acordo com a sua funcionalidade e projeto, sendo que alguns podem analisar um protocolo enquanto outros podem analisar vários (FERREIRA, 2003).

De forma geral, são ferramentas utilizadas em ataques locais em que o invasor deve se situar entre clientes e servidores. O tráfego é monitorado, permitindo a captura de dados e senhas de serviços como FTP e Telnet.

Entretanto este tipo de dispositivo também pode ser utilizado como mecanismo de proteção, monitorando a rede de pacotes estranhos e fora do padrão, sendo de grande utilidade para a segurança de uma rede e auxílio dos administradores.

A necessidade de cuidados e preparação para enfrentar problemas deste nível impulsiona cada vez mais a pesquisa e treinamento na área de segurança. Um exemplo desta preocupação é o desenvolvimento de mecanismos como *firewalls* e sistemas de detecção de intrusão.

3.1.3 **Firewall**

Pode ser qualquer dispositivo com a função de proteger uma ou mais redes de computadores, desde redes que contenham poucas máquinas ou até mesmo com centenas delas. São mecanismos buscam controlar o acesso a redes de computadores, ou seja, eles tentam evitar que pessoas não autorizadas do meio externo acessem a rede interna.

Um *firewall* ao ser instalado em uma rede, pode ser configurado como base de apoio as regras que apresentam as diretivas de acesso de uma organização. Para obter uma maior segurança no ambiente é importante que ele seja utilizado com outros mecanismos de defesa. Segundo Anônimo (2000), as tecnologias de *firewall* podem ser classificadas em três categorias: baseadas em filtragem de pacotes, baseadas em filtragem de pacotes com informação de estado e baseadas em *Proxy*.

Os baseados em filtragem de pacotes normalmente são roteadores com alta capacidade de filtragem de pacotes. Estes para fazer o controle de tráfego da rede baseiam-se nesses dados: endereço de origem, protocolo e número de porta. Por ter uma implementação fácil, faz com que esse tipo seja bastante utilizado, sendo incorporado até dentro de roteadores.

Outro tipo existente é o baseado em filtragem de pacotes com informação de estado, ele pode monitorar o estado das sessões e conexões em tabelas internas e como resultado pode reagir a certos tipos de ataque de negação de serviço. Os produtos baseado nesse tipo de filtragem oferecem proteção a vários recursos específicos de segurança, incluindo correios baseados em *Simple Mail Transfer Protocol* (SMTP) (NORTHCUTT et al., 2002).

O *firewall* baseado em *proxy* funciona da seguinte maneira: quando um usuário remoto se conecta a uma rede, sua conexão passa por uma “peneira” sendo substituída, por

meio de um *proxy*. Com essa técnica, pacotes de IP não são encaminhados diretamente à rede interna, o que ocorre é um tipo de tradução com o *proxy* agindo como condutor e interpretador (FERREIRA, 2003) .

A arquitetura baseada em *proxy* é mais segura que as outras, visto que o *firewall* entende os protocolos de aplicativo como FTP e outros. Porém a desvantagem é que ele exige um envolvimento maior da parte do administrador da rede, pois para cada serviço da rede é necessário um aplicativo *proxy* de ser configurado. Sendo assim, tem-se que os *firewall* baseados nesse modelo são mais lentos.

4 **IDS – INTRUSION DETECTION SYSTEM**

Nos últimos anos a tecnologia de detecção de intrusão (*Intrusion Detection System* - IDS) tem se mostrado uma grande aliada na área da segurança, como fonte de estudos para os administradores da mesma (LAUREANO, 2006).

O termo intrusão pode ser caracterizado como uma violação da política de um sistema. Este tipo de sistema monitora e analisa os eventos de uma rede de computadores, visando encontrar qualquer atividade que comprometa a confiabilidade, integridade e disponibilidade de recursos computacionais ou de rede (NORTHCUTT et al., 2002). Existem dois tipos de IDS: sistemas de detecção de intrusão baseados em rede (*Network-Based Intrusion Detection* – NIDS), e sistemas de detecção de intrusão baseados em *host* (*Host-Based Intrusion Detection* (HIDS).

4.1 SISTEMAS DE DETECÇÃO DE INTRUSÃO BASEADOS EM REDE

Os NIDS são considerados *sniffers* de alto nível, capturando e analisando os pacotes que passam pela rede de forma inativa, ou seja, sem que os outros sistemas percebam isso. Cada pacote capturado na rede é comparado com um conjunto de assinaturas padrões conhecidas. São muito eficientes contra ataques de varredura de portas, falsificação de IP, prevendo ataques a um servidor de Internet e ataques de *buffer overflow* (vulnerabilidade que ocorre quando um programa recebe mais dados do que consegue processar) (FERREIRA, 2003).

Os NIDS têm como pontos positivos:

a) um único IDS pode fornecer monitoramento para múltiplas plataformas;
b) analisam pacotes;
c) monitoram atividades suspeitas em portas conhecidas, como aporta TCP 80, que é utilizada pelo HTTP (*HyperText Transfer Protocol secure*);
d) os ataques podem ser detectados em tempo real e o administrador pode determinar rapidamente o tipo de resposta apropriada;
e) possuem capacidade de detectar não só ataques, mas também tentativas de ataque que não tiveram sucesso;
f) apresentam cuidados para que um *cracker* não possa apagar seus rastros, caso consiga invadir um equipamento;
g) um *cracker* terá dificuldades em saber que existe um NIDS monitorando suas atividades;
h) não causam impacto no desempenho da rede.

Os pontos negativos de um NIDS são:

a) incapacidade de monitorar grandes redes com alto tráfego;

b) dificuldade de compreensão de protocolos de aplicação específicos;

c) não são capazes de monitorar tráfego cifrado;

d) possuem dificuldade de utilização em redes segmentadas, principalmente com *switches*, já que um IDS é um *sniffer*.

4.2 SISTEMAS DE DETECÇÃO DE INTRUSÃO BASEADOS EM *HOST*

Os HIDS possuem componentes que analisam arquivos de registro de eventos do sistema e observam *logins*[3] e processos de usuários. Esses sistemas são bastante diversificados em relação à quantidade de recursos que oferecem, existem os mais avançados que podem capturar instalações de código malicioso, e até terminar execução de processos ilegais (NORTHCUTT et al., 2002).

São funções de um HIDS (FERREIRA, 2003):

a) monitorar acessos e alterações em arquivos importantes do sistema;

b) controlar o uso da Unidade Central de Processamento (UCP);

c) controlar programas em execução;

d) analisar modificações nos privilégios de acesso dos usuários;

e) realizar verificações da integridade dos arquivos do sistema;

f) detectar ataques de força bruta por meio da análise de arquivos de registro de eventos do sistema.

Os sistemas de detecção de intrusão baseados em *host* possuem os seguintes pontos positivos (NAKAMURA; GEUS, 2002):

a) verificação do sucesso ou falha de um ataque, com base nos registros do sistema;

[3] Palavra-Senha ou Palavra-passe

b) monitoramento detalhado das atividades específicas do sistema como acesso a arquivos, modificações em permissões de arquivos, *logon* e *logoff* do usuário e funções do administrador;
c) detecção de ataques que ocorrem fisicamente no servidor;
d) ataques que utilizam criptografia podem passar despercebidos pela rede, mas podem ser descobertos pelo HIDS, pois o sistema operacional antes de mais nada, decifra os pacotes que chegam ao equipamento;
e) é independente de topologia de rede;
f) gera poucos falsos positivos, ou seja, poucos alarmes falsos;
g) não necessita de *hardware* adicional.

Pontos negativos que devem ser considerados dos HIDS:

a) é dependente do sistema operacional, isto é, um HIDS que funciona no Linux é totalmente diferente de outro que opera no Windows;
b) não é capaz de detectar ataques na rede, como por exemplo, a varredura.
c) caso o HIDS for invadido, as informações podem ser perdidas;
d) necessita de espaço de armazenamento adicional para os registros do sistema;
e) apresenta uma baixa de desempenho no computador que está sendo monitorado;
f) não é capaz de emitir alertas em tempo real.

4.3 SISTEMAS DE DETECÇÃO DE INTRUSÃO HÍBRIDOS

Além dos modelos de IDS já comentados, existem os sistemas de detecção de intrusão híbridos. Eles unem as características de NIDSs e HIDSs, estabelecendo uma relação entre os arquivos de registro de eventos e informações de sistema com tráfego de rede. Este

tipo de IDS tenta combinar os aspectos positivos dos sistemas anteriores para que a detecção de intrusão seja mais eficaz.

Esses sistemas operam como se fossem NIDS, capturando e processando pacotes do tráfego e detectando e reagindo a ataques. Porém, efetuam esse processo como HIDS, processando os pacotes endereçados ao próprio sistema. Dessa maneira é possível resolver o problema de desempenho dos IDSs baseados em rede, mas ainda persiste o problema de escalabilidade em sistemas baseados em *host*, sendo que um IDS híbrido deve ser instalado em cada equipamento a ser monitorado (FERREIRA, 2003).

5 EXEMPLOS DE SISTEMAS DE DETECÇÃO DE INTRUSÃO

A área de detecção de intrusão é recente, pois a cada dia são descobertas novas falhas e vulnerabilidades nos sistemas. Logo abaixo será descritos alguns sistemas que tentam contornar esse situação, como o Snort, RealSecure, Intruder Alert e Nuzzler Basic.

5.1 SNORT

É um dos IDSs mais utilizados e populares no momento, que possui código-fonte aberto e pode ser executado em qualquer sistema Unix e Windows. O Snort utiliza um modelo de NIDS, e possui uma ampla base de assinaturas (ataques), de *plug-ins* e aplicativos de suporte (RAITZ, 2005).

Esta ferramenta é suportada em arquiteturas *Reduced Intruction Set Computing* (RISC) e *Complex Intruction Set Computing* (CISC) e em diversas plataformas Linux. Combina eficiência e simplicidade, utilizando uma biblioteca chamada *libpcap* para capturar os pacotes da rede e um analisador simples para tratá-los (FERREIRA, 2003).

Dessa forma, é uma ferramenta IDS considerada avançada, capaz de fazer análise de tráfego em tempo real. A principal função do Snort é detectar ações maliciosas na rede por meio de sua análise de tráfego, e para isso utiliza uma linguagem flexível de regras onde o usuário pode criar outras regras para adequar às suas necessidades, descrevendo qual o tráfego será coletado.

Outro ponto forte dessa ferramenta é o fato de ser leve e pequeno, pois seu código foi desenvolvido na linguagem de programação C e é bastante otimizado, dividindo em módulos, os quais são ferramentas poderosas capazes de produzir uma grande quantidade de informação sobre ataques monitorados (FERREIRA, 2003).

Por apresentar uma base com milhares de assinaturas de ataques, essa ferramenta de detecção de intrusão é considerada bastante completa. Essa base fica disponibilizada na Internet e é utilizada por usuários do mundo inteiro. Segundo Campello e Weber (2001), o Snort possui um conjunto de regras muito semelhante a filtros de rede, embora possua diretivas complexas para a analise e o tratamento dos pacotes coletados.

O Snort é uma boa ferramenta, mas como todo software deve ser configurado da maneira correta. Devem ser aplicadas somente as assinaturas de ataques necessárias a realidade da rede. Além disso, o banco de assinaturas deve estar constantemente atualizado evitando assim que ataques passem despercebidos. O administrador tendo esses cuidados diminui consideravelmente os falsos positivos e falsos negativos.

Este sistema foi projetado para ser executado em muitos sistemas operacionais, entre eles Linux, FreeBSD, NetBSD, OpenBSD, Solaris, Mac e o Windows.

Existem alguns programas opcionais que podem ser instalados para facilitar a administração do sistema. Aqui serão citados alguns deles. Os *softwares* são (SANTOS, 2005):

- MySQL, para armazenamento dos alertas em banco de dados;

- Guardian, para leitura em tempo real dos logs, e bloqueio via firewall;
- Iptables, para bloqueio de ataques;
- Apache, para disponibilização de informações via Web;
- PHP (Personal Home Page) se houver plug-ins que o exija;
- Apache com recursos SSL para monitoramento;
- BASE (*Basic Analysis and Security Engine*), para apresentar de forma mais prática os logs armazenados em banco de dados.

O Snort tem uma arquitetura composta de quatro componentes básicos: o farejador, o pré-processador, o mecanismo de detecção e os plugins[4] de saída. A Figura 8 nos mostra uma visão de alto nível dessa arquitetura.

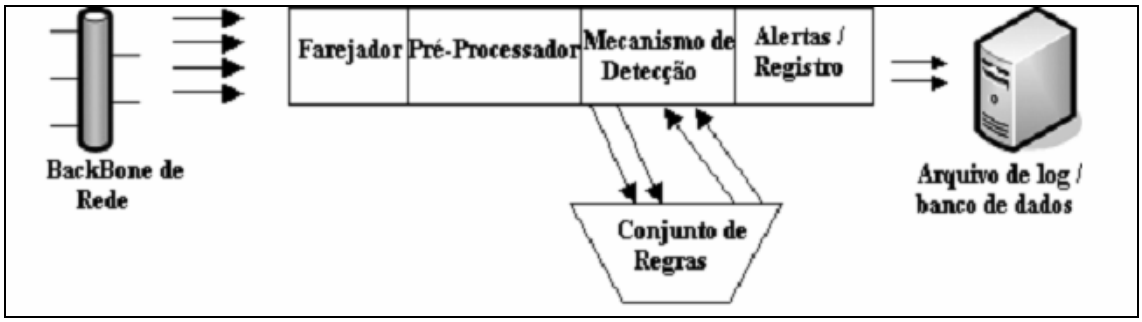

Figura 8. Arquitetura do Snort
Fonte: (SANTOS, 2005).

O Snort utiliza uma linguagem de regras flexível para descrever o tráfego que ele deve analisar ou deixar passar, e também um mecanismo de detecção que utiliza uma arquitetura modular de plugins, que registram os ataques de diversas maneiras, sendo uma delas, registrar os logs em banco de dados (MySQL, PostgreSQL, Oracle, entre outros), em arquivo binário no formato do tcpdump, em arquivo texto ou no syslog.

O processo de detecção é baseado na análise dos pacotes que trafegam na rede, comparando-os com uma base de assinaturas de ataques conhecidos, que são baseadas num

[4] Um programa de computador usado para adicionar funções a outros programas maiores, provendo alguma funcionalidade especial ou muito específica

conjunto de regras, essas são consultadas pelo Snort no momento da análise do pacote, sendo utilizadas de acordo com a definição no arquivo de configuração (SANTOS, 2005).

As regras já acompanham o IDS, sendo necessário ao administrador mantenha-las sempre atualizadas, entretanto, ainda é possível escrever novas regras ou alterar as já existentes de forma a acrescentar mais funcionalidades a elas.

Segundo SANTOS (2005), o Snort pode monitorar diferentes pontos da rede ao mesmo tempo. Entre esses pontos podemos citar:

- Normalizar requisições HTTP;
- Detectar ataques do tipo Unicode;
- Detectar Buffer Overflow;
- Detectar portscan;
- Remontar os segmentos TCP;
- Ativar regras dinamicamente (regras podem ser ativadas por outras regras)

5.1.1 **Componentes do Snort**

A base do Snort está montada em cima da biblioteca Libcap. Esta provê funções de acesso a recursos de rede de baixo nível, como monitoramento do sistema, debugging de rede entre outros. A Figura 9 nos mostra os componentes do Snort e oferece uma visão de alto nível.

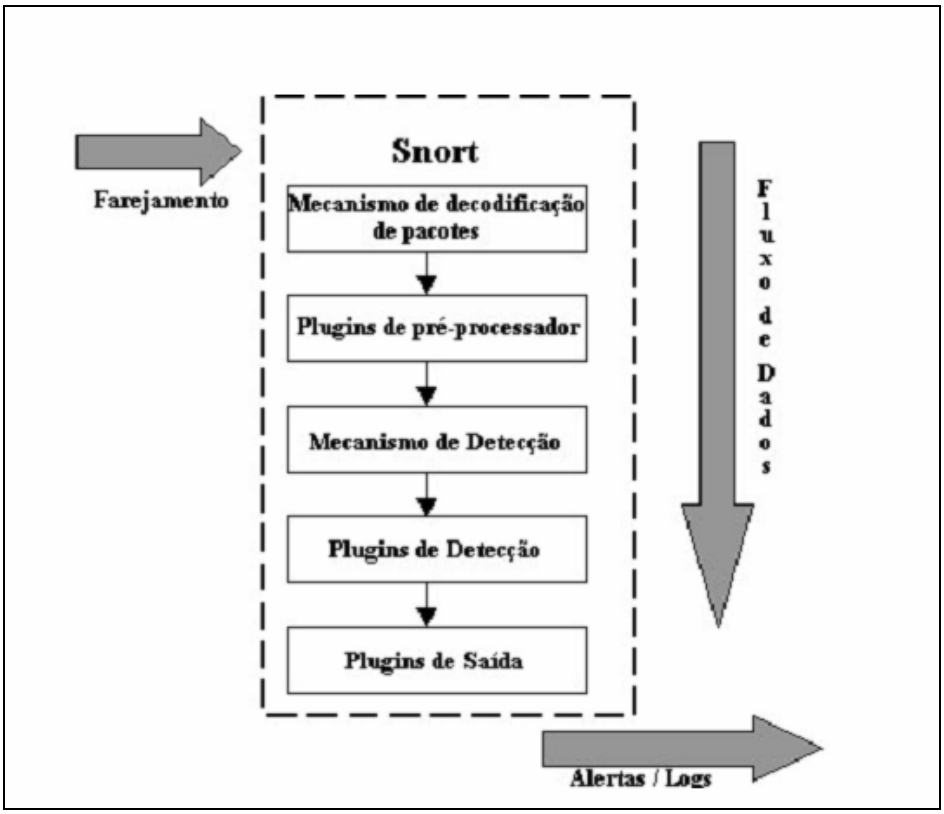

Figura 9. Componentes do Snort
Fonte: Fonte: (SANTOS, 2005).

A seguir apresenta-se os componentes do Snort, e suas funções dentro do processo de detecção de invasões:

1. **Mecanismo de captura** – o tráfego é obtido da rede, através da biblioteca Libpcap. Os pacotes são encaminhados para o mecanismo de detecção, que monta a estrutura dos pacotes para os protocolos de enlace, sendo esses ainda mais decodificados para os protocolos de nível mais alto.
2. **Plugins de pré-processadores** – Os pacotes são examinados e manipulados antes de serem enviados ao mecanismo de detecção. Cada pré-processador analisa se esse pacote é algo que ele deve examinar, alertar ou modificar.
3. **Mecanismo de detecção** – verifica cada pacote em relação às opções listadas no arquivo de regras. Cada uma das opções de palavra-chave da regra é vinculada a um plugin de detecção que pode realizar outros testes.

4. **Plugins de saída** – é a saída dos alertas do mecanismo de detecção, dos pré-processadores ou do mecanismo de decodificação.

A Figura 10 usa a pilha de protocolos TCP/IP e ilustra a atuação dos componentes do Snort.

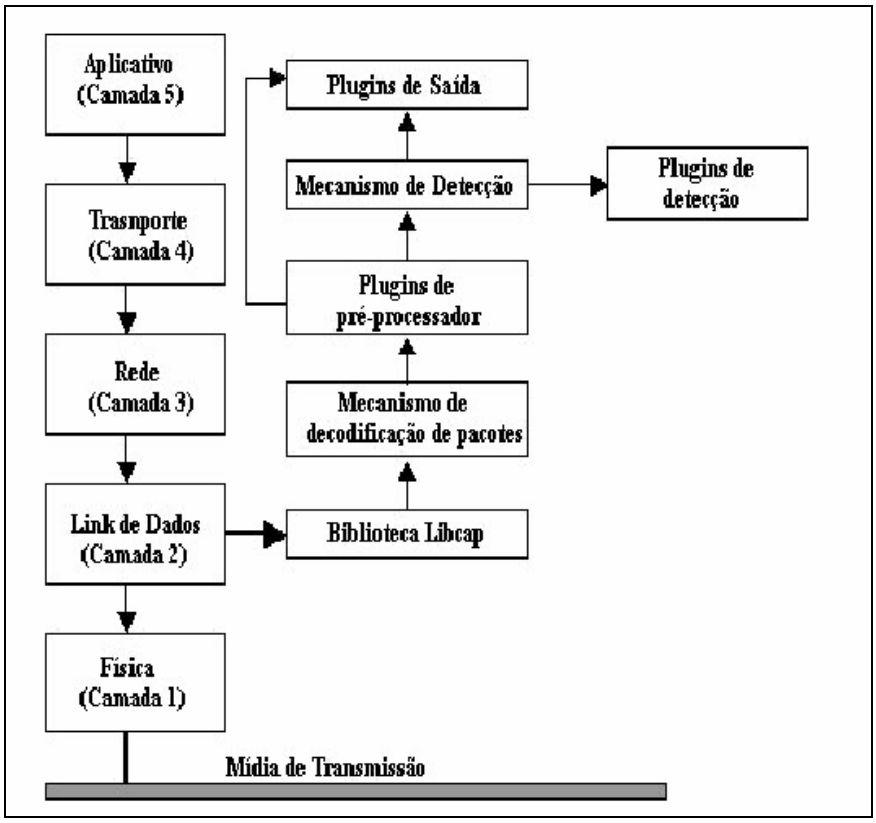

Figura 10. O Snort e o modelo TCP/IP
Fonte: (SANTOS, 2005)

5.1.2 **Mecanismo de Captura**

O comportamento padrão de uma placa de rede é ignorar o tráfego que não é destinado para ela. É necessário mudar este comportamento, fazendo com que a mesma verifique todo o tráfego do segmento de rede ao qual ela se encontra. Para isso a placa deve operar em modo promíscuo.

O mecanismo utilizado para colocar esta placa no modo de operação promíscuo é a biblioteca LibPcap. Esta biblioteca foi escrita como parte de um programa maior chamado

TCPdump. Esta biblioteca permite que desenvolvedores executem códigos para receber pacotes da camada de link de dados em diferentes sistemas operacionais, sem se preocupar com as características ou restrições das placas de redes e drivers (SANTOS, 2005).

A biblioteca LibPcap pega pacotes diretamente da placa de rede, funcionando de maneira semelhante a uma escuta telefônica. Permitindo assim, que um aplicativo ou um dispositivo de hardware se "intrometa" no tráfego da rede de dados.

5.1.3 **Pré-processadores**

Depois de capturados, os pacotes são direcionados para os pré-processadores. A idéia principal por trás disso, é fornecer uma estrutura para permitir alertar, eliminar e modificar os pacotes, antes que eles cheguem ao mecanismo de detecção principal.

Os recursos de detecção do Snort ocorrem através da análise dos dados dos pacotes com suas regras, que é uma evolução das assinaturas (SANTOS, 2005).

As assinaturas são especificações de ataques através de correspondência de número e string em relação a parte particular do pacote. São implementados recursos como a detecção de anomalias de protocolos, através de pré-processadores. Estes manipulam os dados dos pacotes após o decodificador ter analisado os campos do pacote, mas antes que o mecanismo de detecção comece a fazer comparações com regras.

Existe um custo na inclusão de pré-processadores. A velocidade do Snort é derivada de sua base de correspondência de regras simples. Sempre que um pré-processador for acrescentado haverá perda de rendimento. O grau de perda de performance está ligado com a forma de implementação dos pré-processadores. A implementação se dá por meio de plugins modulares no Snort, onde se pode decidir exatamente quais e quantos pré-processadores serão ativados. O pré-processador recebe os pacotes capturados e os verifica

em relação aos plugins. Estes verificam certos padrões dos pacotes. Após a detecção do padrão do pacote, ele é encaminhado para o mecanismo de detecção. Este IDS possui plugins de RPC (*Remote Procedure Call*), IIS (*Internet Information Services*), Telnet, de fragmentação, entre outros.

Os plugins possuem uma importante característica dentro do Snort, sendo que podem ser desabilitados e ativados conforme a necessidade. Os pré-processadores facilitam a escrita de regras, diminuem a presença de falso positivo / negativo, enquanto mantém o desempenho. Alguns objetivos para os quais os pré-processadores são utilizados (SANTOS, 2005):

- Remontagem de pacotes;
- Decodificação de protocolos;
- Detecção não baseada em regras ou baseada em anomalias.

São utilizados quando queremos detectar algo que a detecção baseada em regra direta por si não consegue.

5.1.4 **Mecanismo de Detecção**

Depois dos pacotes serem capturados, eles tem que passar por um processo de decodificação, e são colocados nas estruturas de dados, organizando, filtrando e decodificando o fluxo de dados.

Segundo SANTOS (2005), este mecanismo pode ser considerado a parte principal do Snort. Ele recebe os dados e verifica através de um conjunto de regras (assinaturas), que são baseadas em texto. Os arquivos de regras são classificados em diferentes grupos, como por exemplo, o arquivo ftp.rules que contém uma lista de ataques e explorações FTP.

Quando iniciado o sistema Snort lê todos os arquivos de regras e cria uma lista encadeada, sendo que mais tarde será utilizado dessa lista para fazer as comparações necessárias aos pacotes.

Existem cinco encadeamentos de regras separados, que na verdade são os cabeçalhos da lista, esses podem ter os seguintes valores (SANTOS, 2005):

- Activation: Alertar e ativar outra regra;
- Dynamic: Registra o tráfego quando chamado por outra regra;
- Alert: Gera um alerta e depois registra o pacote;
- Pass: Ignora esse pacote;
- Log: Registra o tráfego.

Entretanto para cada um dos cinco encadeamento, existem listas encadeadas separadas, divididas por protocolo. Esse nível da árvore é citado como RTN (*Rule Tree Nodes*). Os quatro protocolos suportados são (SANTOS, 2005):

- TCP;
- UDP;
- ICMP;
- IP.

Dentro de cada uma das listas encadeadas de protocolo estão às opções da regra, que são mencionadas como OTN (*Options Tree Nodes*). Um exemplo seria:

- *Content*: Conteúdo verificado pelo algoritmo de correspondência de padrões;
- *Flow*: *Link* para plugin de detecção.

Conforme já informado anteriormente na inicialização do Snort é feito o carregamento das regras e preenchimento das listas encadeadas. Após isso, é necessário um método de navegação para procurar uma correspondência com os pacotes. Quando o pacote e

verificado pelo mecanismo de detecção, o Snort analisa os cabeçalhos de regra seguindo essa ordem: *Activation, Dynamic, Alert, Pass e Log*.

Dentro de cada cabeçalho de regras, os RTN e OTN serão verificados. Quando o Snort encontra uma correspondência, o algoritmo percorre as colunas debaixo, procurando uma correspondência dentro de cada um dos OTN's. Por exemplo, quando é encontrada uma correspondência de exploração do próprio Snort, segue-se da seguinte forma, dentro do nó de opção existem dois itens (SANTOS, 2005):

- Um ponteiro para um plugin de detecção. Este verifica se o pacote corresponde a uma sessão estabelecida;
- O padrão que se procura. Usa-se o algoritmo de pesquisa rápida de *string Boyer-Moore*.

Após encontrar a correspondência correta, o processo de busca sai da estrutura em árvore e retorna ao cabeçalho *Alert.* Neste momento será gerado o alerta e o processo se encerra.

O Snort utiliza uma estratégia de saída muito rápida, pois quando uma correspondência de um pacote é encontrada em alguma regra, não se verifica mais esse pacote com nenhuma outra.

5.1.5 **Plugins de Saída**

São responsáveis por realizar um procedimento, que pode ser de gerarem alertas ou tomar alguma medida em imediato. A Figura 11 nos mostra os procedimentos que podem ser adotados por esses plugins.

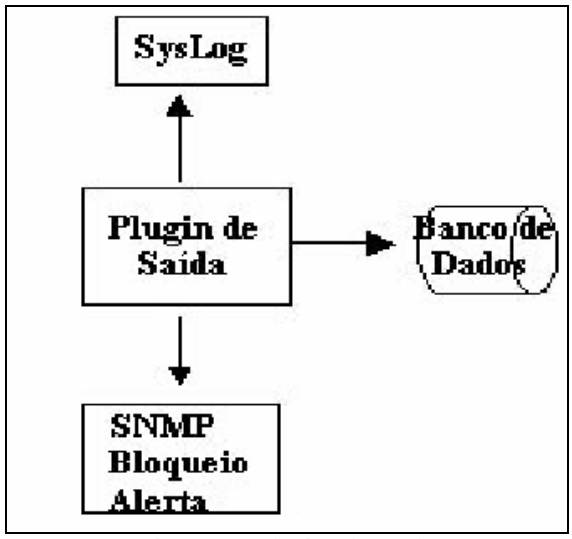

Figura 11. Plugins de Saída
Fonte: (SANTOS, 2005)

Os plugins de saída podem ser configurados para "conversar" com *firewall*, podendo assim enviar alertas via e-mail, gravar em arquivos textos, MySQL entre outros.

É importante ressaltar os plugins Snortsam e o Guardian. Esses dois tem a capacidade de gerar regras de *firewall*, bloqueando tentativas de invasão em tempo real (SANTOS, 2005).

O Snortsam oferece uma ligação entre a detecção de uma exploração e a configuração de um *firewall,* que pode ser utilizado para bloquear o endereço IP de origem do ataque.

O Guardian é uma ferramenta que tem como principal característica a capacidade de ler os *logs* do Snort em tempo real e gerar regras para o *firewall* bloqueando os IP's de origens dos ataques.

5.2 REALSECURE

Essa ferramenta utiliza uma arquitetura híbrida, combinando as vantagens de análises baseadas em rede com as baseadas em *host*. O RealSecure é baseado em dois componentes: sensores e gerentes. O primeiro é responsáveis pelas tarefas de geração e

análise de dados, já o segundo é um componente usado para tarefas administrativas e operacionais (FERREIRA, 2003).

Os sensores estão divididos em: rede e *host*. Os de rede analisam os pacotes que atravessam um segmento específico e respondem a possíveis intrusões. As respostas podem ocorrer com o término de uma conexão, gravando a sessão, reconfigurando *firewalls* ou tomando alguma outra ação definida pelo usuário. Além disso, é enviando um alerta aos módulos de gerência, relatando o caso detectado (CAMPELLO e WEBER, 2001).

Os Sensores baseados em *host* complementam a análise de rede utilizando trilhas de auditoria em formatos específicos de acordo com o sistema operacional utilizado (CAMPELLO e WEBER, 2001).

5.3 INTRUDER ALERT

Esta ferramenta foi desenvolvida pela Symantec e utiliza um modelo de HIDS e gerenciamento de políticas de segurança. Possibilita aos administradores uma monitoração em tempo real. Ele também permite que sejam rapidamente criadas e aplicadas regras, fornecendo de imediato atualizações de detecção de intrusão em formato de tabelas e gráficos. Esse IDS possui agentes de *software* especializados que suportam a maior parte das plataformas de serviços como Windows NT e versões comerciais do Unix (FERREIRA, 2003).

A instalação e o ajuste remoto tornam fácil a distribuição do *software* e a manutenção do sistema, além de não interferir nas tarefas de rotina executadas pelos usuários. Quanto às ações a serem tomadas devido a uma ameaça à segurança, esta ferramenta dispara um alarme ou dependendo da situação encaminha o evento ao console apropriado para tomar as devidas providências.

5.4 NUZZLER BASIC

Esse IDS permite localizar pacotes de dados suspeitos e possibilita a análise de redes à procura de vírus e *trojans*[5]. Uma das vantagens desse sistema é a possibilidade de ser operado de qualquer ponto da rede. Sua interface permite uma pesquisa rápida referente a todas as funções e elementos de exibição importantes (FERREIRA, 2003).

O Nuzzler Basic disponibiliza uma biblioteca de regras e assinaturas de ataques, sendo que novas regras poderão ser criadas e editadas a qualquer momento pelo próprio operador do sistema. Essa ferramenta é de distribuição gratuita.

Algumas vantagens dessa ferramenta (FERREIRA, 2003):

a) interface fácil para avaliação rápida dos arquivos de registro de eventos;

b) analisa mais de 1.000 pacotes de dados em poucos segundos;

c) o monitor de tráfego pode mostrar os dados que trafegam na rede;

d) possui janela de regras temporárias para as próprias regras;

e) possui filtragem de pacotes avançada;

f) essa ferramenta executa na maioria das plataformas Windows.

[5] *Trojan* é um programa criado para obter informações do usuário como senhas, por exemplo.

6 TRABALHOS CORRELATOS

6.1 PROTEÇÃO DE DETECTORES DE INTRUSÃO ATRAVÉS DE MÁQUINAS VIRTUAIS

Esse artigo foi apresentado por Laureano, Maziero e Jamhour. Nele foi descrito uma forma de aumentar a segurança de sistemas computacionais utilizando de máquinas virtuais. A proposta usava o isolamento de espaços de execução providos pela maquina virtual para separar o detector de intrusão do sistema a monitorar. Sendo assim o detector ficava instalado no *host* e analisava os dados enviados ao ambiente virtual ao qual estava configurado a verificar. Os resultados mostraram viabilidade nessa solução.

6.2 UTILIZAÇÃO DE MÁQUINAS VIRTUAIS PARA IMPLANTAR UM MECANISMO TRANSPARENTE DE DETECÇÃO DE INTRUSÃO EM SERVIDORES WEB

Essa monografia foi apresentada por Luciano Raitz, em sua especialização na FURB. O objetivo deste trabalho foi montar uma arquitetura confiável para o uso de detectores de intrusão através de maquinas virtuais. A arquitetura proposta faz uso da maquina virtual para deixar o sistema IDS inacessível e invisível, caso haja uma invasão. Os testes mostram que essa arquitetura e funcional e viável para servidores.

6.3 DETECTANDO INTRUSÕES NA MÁQUINA VIRTUAL USER-MODE LINUX

O artigo “Detectando Intrusões na Máquina Virtual User-Mode Linux” foi apresentado por apresentado Laureano, Maziero e Jamhour.

Neste trabalho o objetivo era restringir a execução de processos suspeitos na máquina virtual e conseqüentemente evitar o comprometimento do sistema. Os testes mostram a eficiência da arquitetura.

7 IMPLEMENTAÇÃO DE UM SISTEMA IDS EM UMA MÁQUINA VIRTUAL

A implementação de um sistema IDS em um ambiente virtual foi realizada com uma simulação muito próxima do que poderia acontecer em um determinado caso real.

A arquitetura desenvolvida nesse trabalho, demonstrar uma maneira segura para proteger o IDS através do uso de máquina virtual. O sistema operacional a ser protegido juntamente com seus *softwares* deve ser executado no ambiente virtual com sua configuração padrão como se estivesse em um *Hardware* comum.

7.1 ESCOLHA DOS SOFTWARES UTILIZADOS NOS TESTES

Existem vários motivos para a escolha do sistema operacional Windows XP Profissional, sendo algumas delas a facilidade de acesso a documentações, a interoperabilidade do sistema Windows de trabalhar referente a heterogeneidade de sistemas existentes no mercado hoje, entre outras.

O VirtualBox é um programa utilizado para a virtualização de sistemas operacionais, foi utilizado para o ambiente de testes ser implementado. O computador onde foi instalado a máquina virtual, é um notebook com o sistema operacional Windows XP Profissional, e dentro desse ambiente, 2 (duas) máquinas virtuais, uma com o sistema Windows e outra com Linux, a fim de se fazer um ambiente o mais próximo possível do real, onde as máquinas são representadas como sendo normais de produção a uma determinada situação, assim como também no computador de onde foram feitos os ataques, que tem o sistema operacional Windows XP Profissional.

No ambiente virtual foi instalado o IDS Snort, pois foram feitas varias pesquisas e observou-se que este sistema de detecção é hoje um dos mais requisitado, estudado e amplamente atualizado.

7.2 ESTUDO DE CASO: MÁQUINAS VIRTUAIS NA PROTEÇÃO DE SISTEMAS DE DETECÇÃO DE INTRUSÃO

Para que se possa desenvolver a proposta, foi utilizada a seguinte arquitetura:

a) Sistema operacional Windows XP Professional (Service Pack 3);

b) IDS Snort (Versão2_8_6_1);

d) MySql (Versão 5.1.50);

e) Apache (Versão 2.2.16);

f) Basic Analysis and Security Engine (BASE) (Versão 1.4.5);

g) Máquina Virtual VirtualBox (Versão 3.2.8).

Para a realização dos testes foi utilizado de um modem roteador e de um *hub*, conforme pode ser observado na Figura 12:

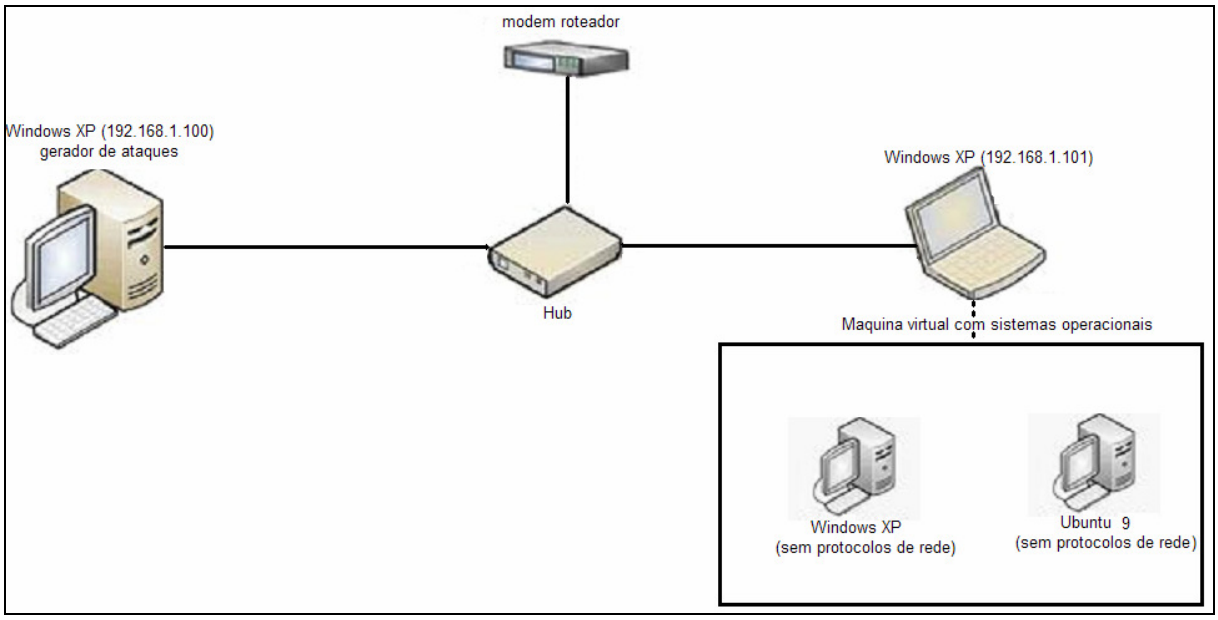

Figura 12. Ambiente montado para realização dos testes
Fonte: Do autor.

Conforme pode ser observado na Figura 13, tanto no sistema *host* como no sistema anfitrião foi instalado o sistema Windows XP Profissional. No VirtualBox, além do sistema convidado que será utilizado para mostrar os testes, também existe outra máquina virtual, com o sistema Linux Ubuntu 9.

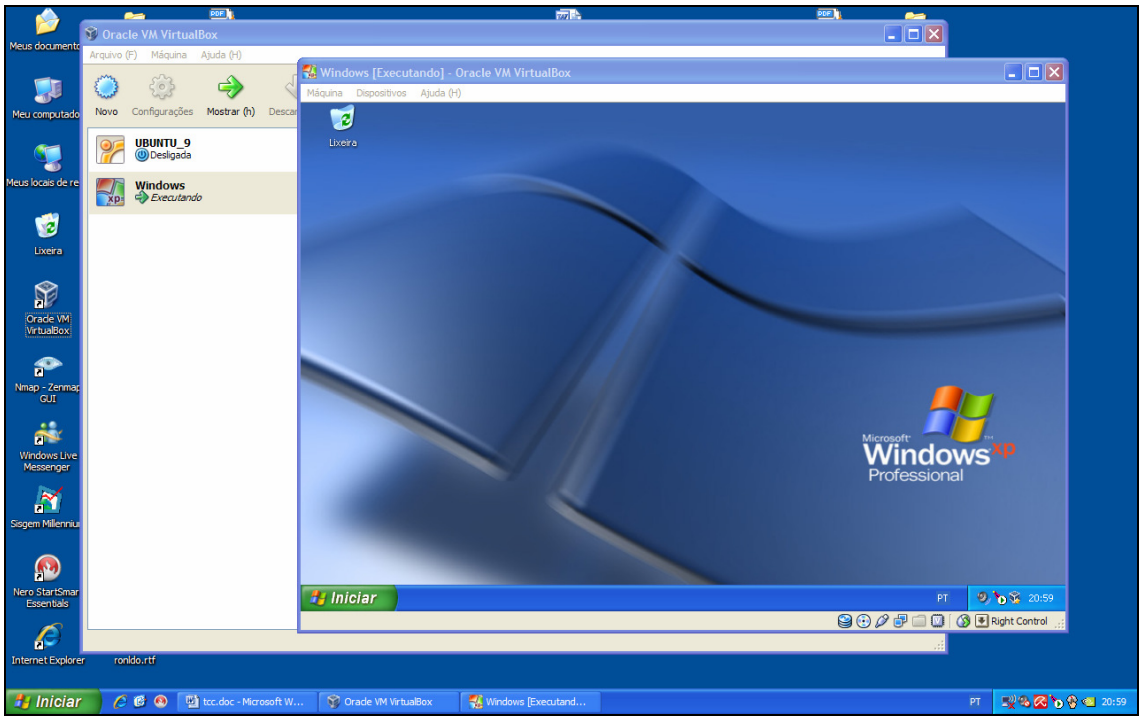

Figura 13. Sistemas operacionais
Fonte: Do autor

O ambiente virtual foi criado usando o método de utilização de discos virtuais, pois assim não se tem a necessidade de ter que criar partições no disco rígido, facilitando a instalação do mesmo e até sua cópia.

A configuração de rede deve ser modificada para que a arquitetura proposta tenha o funcionamento correto. De maneira que o sistema virtualizado não tenha protocolos de rede ativos, tornando-o inacessível e protegido de qualquer ataque via rede. Porém, é necessário criar uma conexão direta a rede local, sendo que essa alteração é feita através do VirtualBox antes da inicialização do mesmo, para que assim o IDS instalado no sistema convidado tenha acesso aos dados que trafegam no ambiente não virtualizado, pois por padrão uma máquina

virtual cria sua própria rede, e torna impossível a análise das informações direcionadas a rede local. Essa alteração pode ser observada na Figura 14.

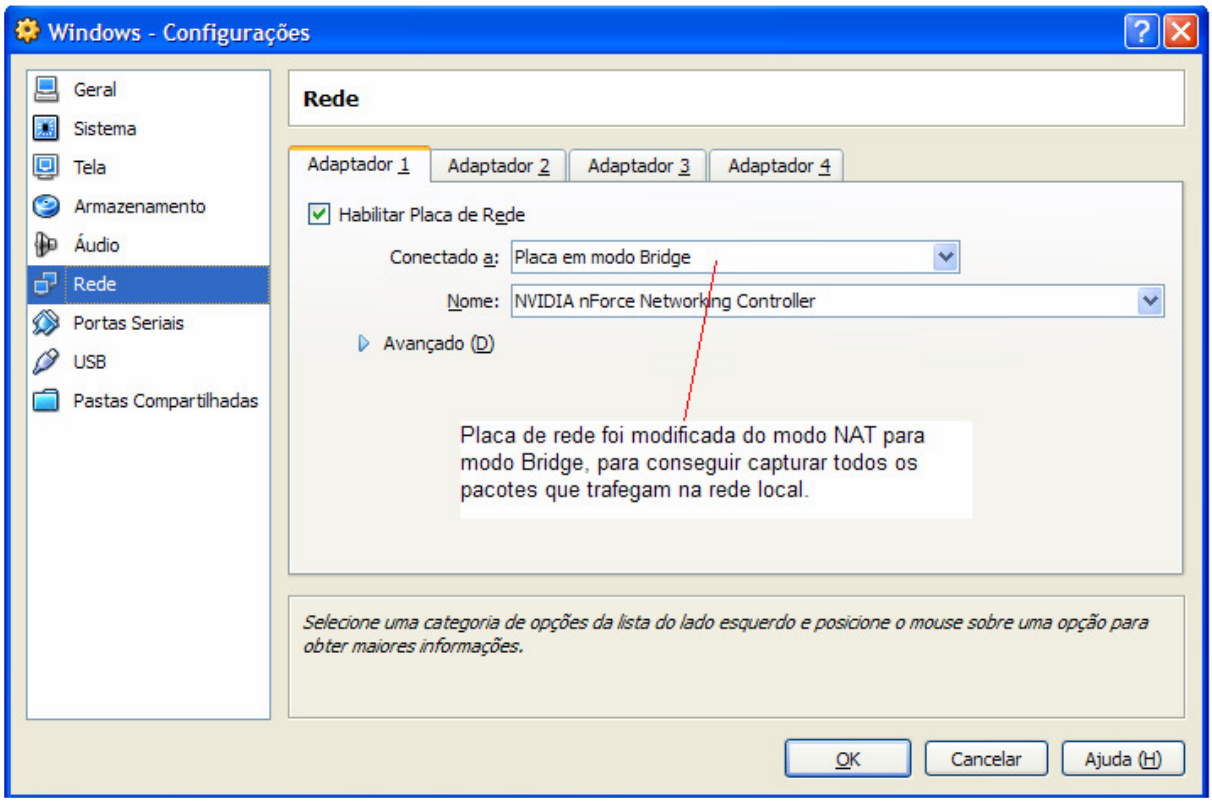

Figura 14. Alteração da placa de rede do ambiente virtual
Fonte: Do autor.

Após a montagem da arquitetura, foi instalado o IDS Snort no sistema convidado, sendo configurado para capturar todos os pacotes que tenham como destino ou origem *hosts* da rede em análise. Para que isso aconteça é necessário que sejam feitas alterações no arquivo "snort.conf", arquivo esse que contém a configuração do Snort. A seguir parte da configuração, onde foi definida qual rede ele deve analisar (mais detalhes no apêndice A).

```
#var HOME_NET any

var HOME_NET 192.168.1.0/24
```

Posteriormente o MySql foi instalado para que os pacotes capturados fossem gravados no banco de dados, e assim disponibilizando-os para futuras consultas para análise dessas informações. Além desse banco de dados, o Snort permite também trabalhar com outros bancos de dados, como o PostgreSQL e Oracle.

Em seguida o Apache foi instalado, permitindo assim o uso da ferramenta BASE, pois a mesma necessita estar instalada em um servidor *Web* para funcionar.

O BASE foi a ferramenta utilizada para a consulta dos pacotes capturados. A seguir a Figura 15 nos mostra a tela principal, que mostra os número de alertas, protocolos, portas de origem e de destino entre outras informações.

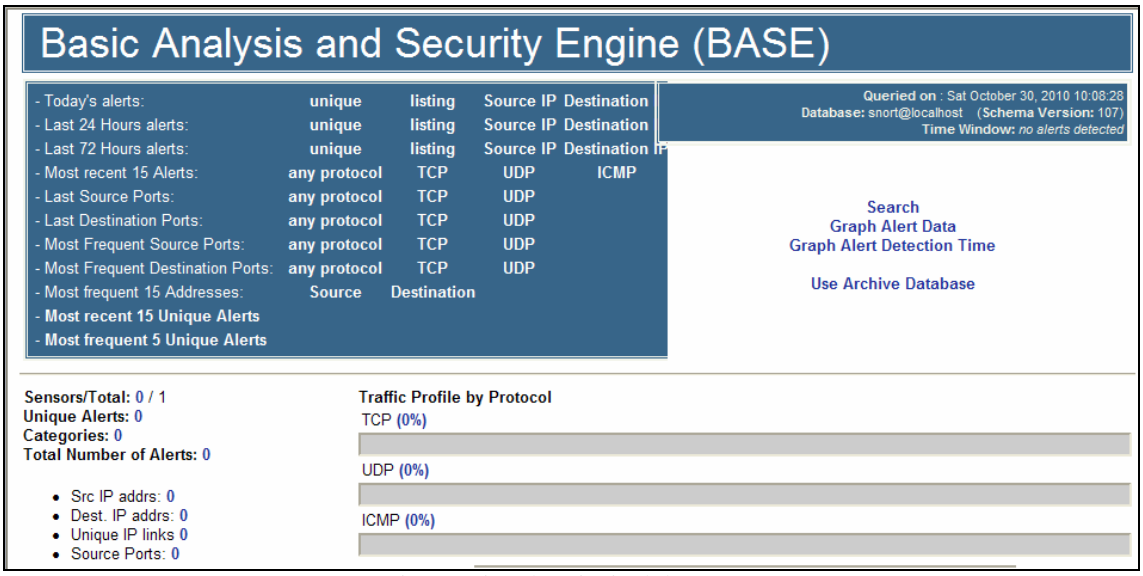

Figura 15. Tela principal do BASE
Fonte: Do autor

Neste *software* basta clicar na informação desejada, e obter mais detalhes sobre a mesma, como por exemplo, cada pacote capturado pode ser exibido em um formato decodificado. O BASE também permite outras consultas, como por protocolo ou até mesmo por categoria dos pacotes.

Para a realização dos testes foram utilizadas as ferramentas Nmap e GFI LANguard. O Nmap é *software* livre utilizado para realizar uma varredura de portas e serviços, sendo muito eficiente para avaliar a segurança de um computador, buscando detectar quais serviços estão rodando no mesmo. O GFI LANguard também é um scanner, que procura falhas de segurança emitindo um relatório ao final informando quais pontos estão com falhas.

7.3 REALIZAÇÃO DE TESTES NA MÁQUINA VIRTUAL

Após a instalação e configuração de todos os *softwares*, inicia-se a partes de testes. Primeiramente inicia-se o Snort conforme Figura 16 nos mostra, através de um comando no *prompt* de comando.

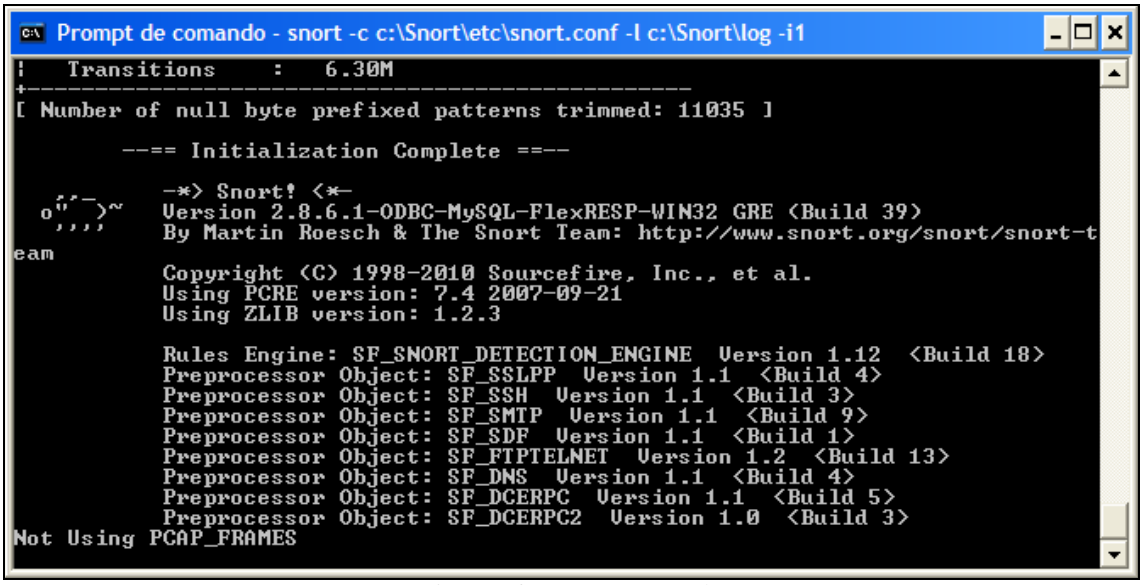

Figura 16. Snort executando
Fonte: Do autor.

O total de pacotes processados é grande, mas apenas parte desses pacotes serão armazenados, pois os outros pacotes foram descartados pelas regras de assinatura do Snort. Existe também a possibilidade de trabalhar em modo promíscuo, onde tudo que passa na rede além de ser analisado, seria também armazenado, porém isso causaria um volume de dados muito alto, além de não ser necessário, pois muitos desses pacotes não ofereceriam risco algum.

Para o bom funcionamento de um IDS, é recomendando que nunca seja desativado o serviço, pois uma vez desativado, pode ocorrer uma invasão na rede e não ser percebido. Para mostrar que o sistema está capturando e analisando os pacotes que trafegam na rede a qual ele está configurado para analisar, foi feito uma varredura de portas (espécie de ataque via rede) a um *host* da mesma rede, tendo como IP “192.168.1.100”, e o *host* que gerava a varredura com o IP “192.168.1.101”. A Figura 17 nos mostra o exemplo:

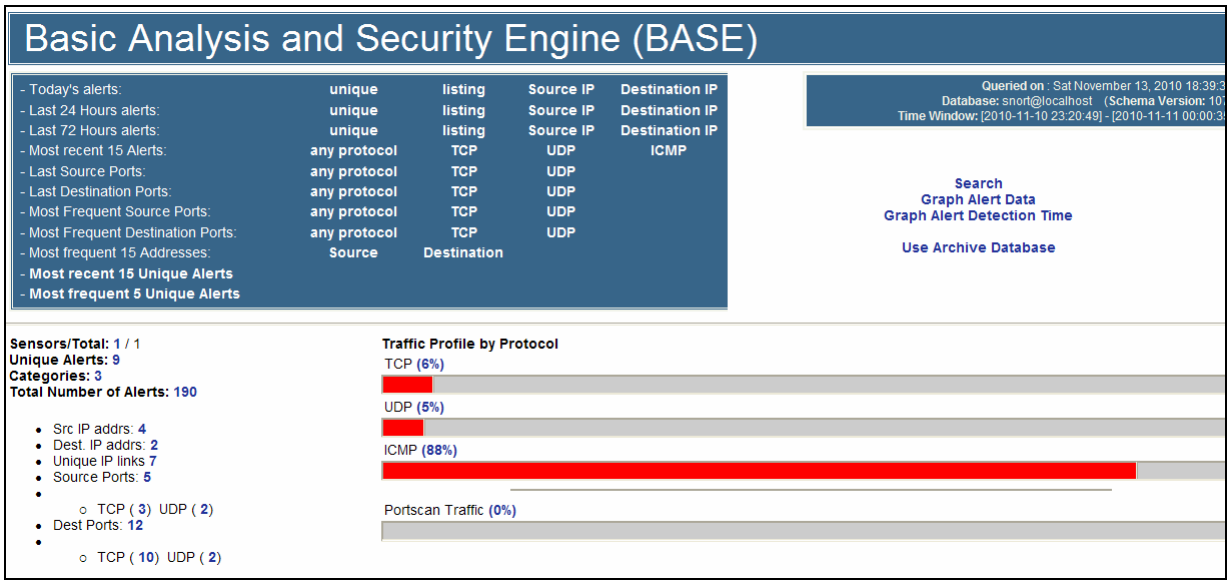

Figura 17. Resultados após varredura de portas – interface principal
Fonte: Do autor

Nesta interface do BASE pode-se observar a quantidade de alertas encontrados, a categoria em que eles se encontram (TCP , UDP e ICMP), de qual IP foram enviados e qual o destino. Todos os alertas aqui ilustrados estão armazenados em um banco de dados, provendo assim a possibilidade de futuramente o administrador consulta-los e analisar-los. Esta ferramenta também nos permite maiores detalhes dos alertas, bastando clicar sobre a categoria ou classe de interesse, na Figura 18 podemos visualizar um bom exemplo:

#46-(1-39)	[local] [snort] ATTACK-RESPONSES 403 Forbidden	2010-11-10 23:58:42	192.168.1.100:80	192.168.1.101:1085	TCP
#47-(1-38)	[local] [snort] ATTACK-RESPONSES 403 Forbidden	2010-11-10 23:58:42	192.168.1.100:80	192.168.1.101:1084	TCP
#48-(1-37)	[cve] [icat] [cve] [icat] [local] [snort] ICMP Destination Unreachable Fragmentation Needed and DF bit was set	2010-11-10 23:58:37	10.1.1.1	192.168.1.100	ICMP
#49-(1-30)	[local] [snort] ICMP PING undefined code	2010-11-10 23:58:31	192.168.1.101	192.168.1.100	ICMP
#50-(1-31)	[local] [snort] ICMP Echo Reply	2010-11-10 23:58:31	192.168.1.100	192.168.1.101	ICMP
#51-(1-32)	[local] [snort] ICMP PING	2010-11-10 23:58:31	192.168.1.101	192.168.1.100	ICMP
#52-(1-33)	[local] [snort] ICMP Echo Reply	2010-11-10 23:58:31	192.168.1.100	192.168.1.101	ICMP
#53-(1-34)	[local] [snort] SHELLCODE x86 inc ebx NOOP	2010-11-10 23:58:31	192.168.1.101:49415	192.168.1.100:40534	UDP
#54-(1-35)	[cve] [icat] [cve] [icat] [local] [snort] ICMP Destination Unreachable Port Unreachable	2010-11-10 23:58:31	192.168.1.100	192.168.1.101	ICMP
#55-(1-36)	[local] [snort] SHELLCODE x86 inc ebx NOOP	2010-11-10 23:58:31	192.168.1.100	192.168.1.101	ICMP
#56-(1-23)	[local] [snort] ICMP PING undefined code	2010-11-10 23:58:29	192.168.1.101	192.168.1.100	ICMP
#57-(1-	[local] [snort] ICMP Echo Reply	2010-11-10	192.168.1.100	192.168.1.101	ICMP

Figura 18. Resultado parcial após varredura (detalhes)
Fonte: Do autor

Esta interface nos trás detalhes de todos os alertas encontrados até o momento, nos mostrando qual o tipo, data, IP de destino e de origem e a qual categoria pertence.

Foi observado que mesmo após ter sido configurado o sistema convidado sem nenhum protocolo de rede, o Snort consegue analisar todo o tráfego, possibilitando assim verificar que está ocorrendo algum ataque este pode ser observado para analise, visto na imagem acima.

7.4 REALIZAÇÃO DE TESTES NO SISTEMA HOSPEDEIRO

Após ter sido feito testes com o sistema IDS dentro da máquina virtual e o mesmo obtendo sucesso na análise de todo o tráfego de rede, foram feitos agora testes no sistema hospedeiro (ambiente real) para tentar encontrar algum rastro de que o mesmo tivesse um ambiente virtual ou sistema de detecção de intrusão em execução.

Primeiro foi feito uma varredura de ataques com a ferramenta Nmap, onde é informado o IP (192.168.1.100 – máquina real) ao qual se deseja fazer um *scan* (alvo), veja os resultados na Figura 19:

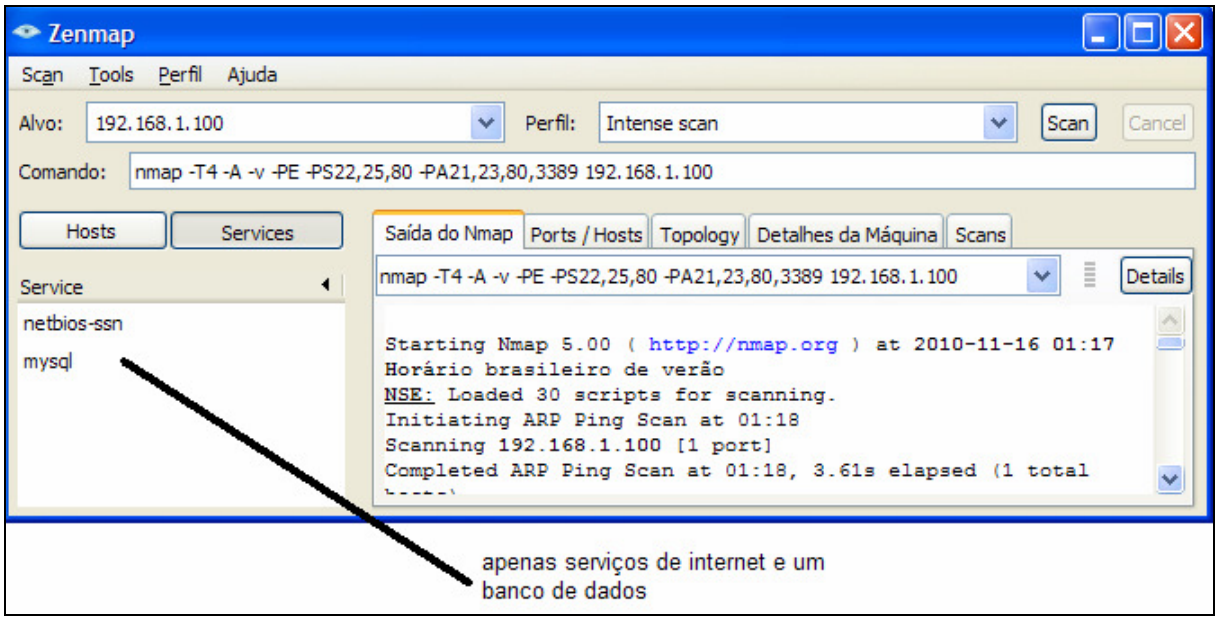

Figura 19. Leitura de serviços em execução
Fonte: Do autor.

O *software* Nmap fez um Scan completo na máquina alvo e encontrou apenas serviços de rede do Windows e um banco de dados executando no momento. A Figura 20 nos mostra com mais detalhes essa varredura.

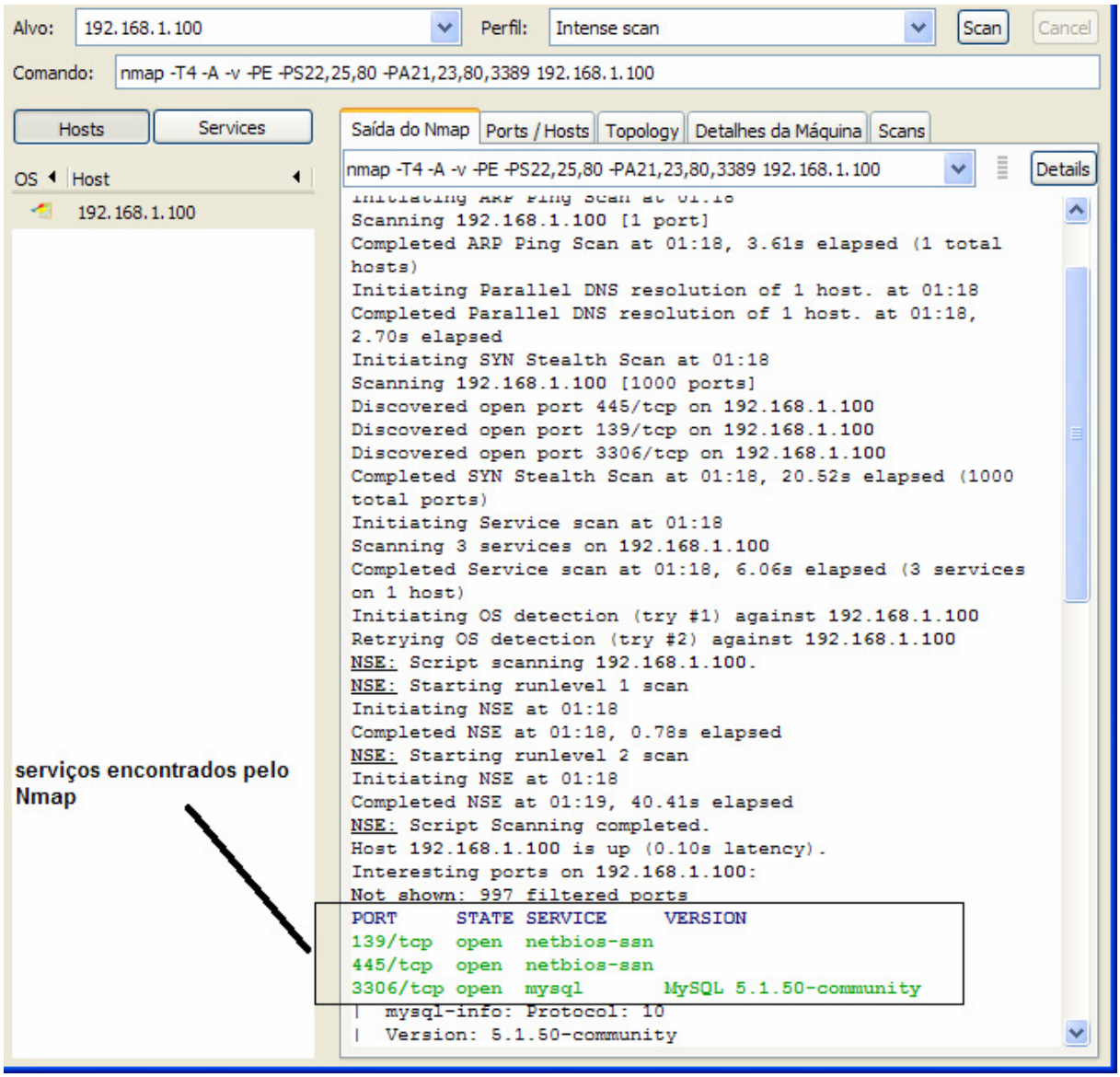

Figura 20. Varredura detalhada
Fonte: Do autor.

Foram efetuadas diversas varreduras e nenhuma delas apresentou informação sobre IDS ou máquina virtual sendo executada, sendo que no momento do *scan* o *host* em análise estava com o ambiente virtual em plena execução e o IDS capturando informações da rede. Segue agora varredura executada pela ferramenta GFI LANguard, onde foi indicado o mesmo IP, a Figura 21 nos mostra o resultado dos processos listados em execução, e a Figura 22 nos traz todos os serviços encontrados.

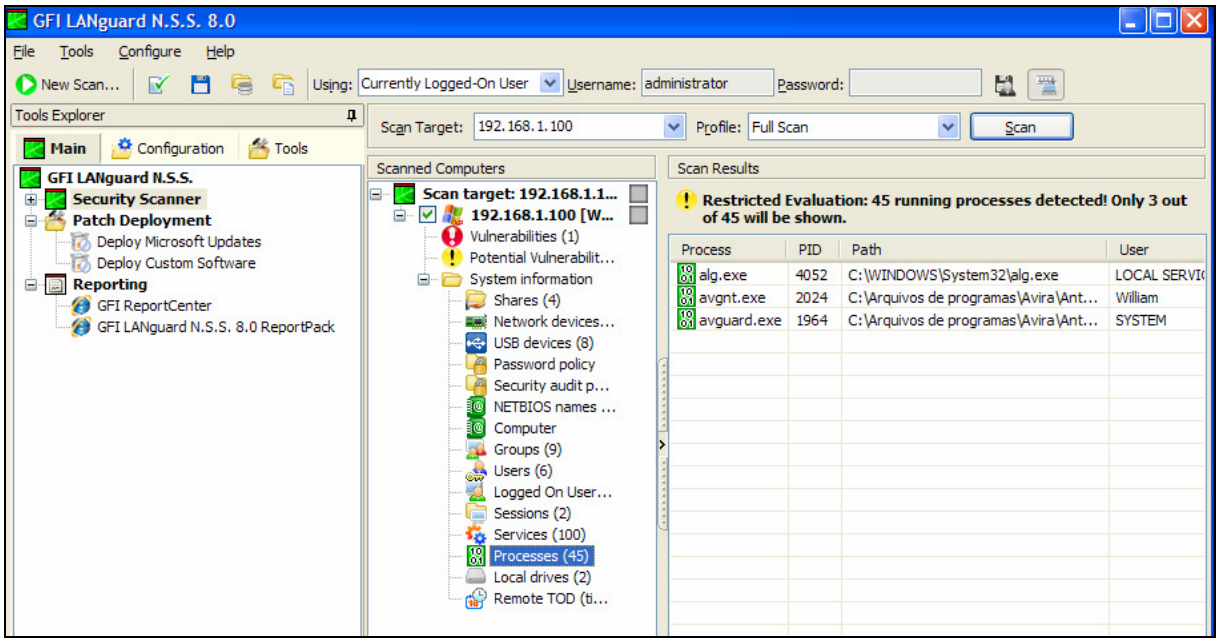

Figura 21. Processos encontrados em execução
Fonte: Do autor.

A interface do GFI LANguard nos mostra os processos que o *software* encontrou, podemos citar como exemplo o antivírus "avgnt.exe". Esta ferramenta é bem completa, ela lista todos os periféricos de *hardware* da máquina, serviços, ambientes compartilhados, usuários, entre outros.

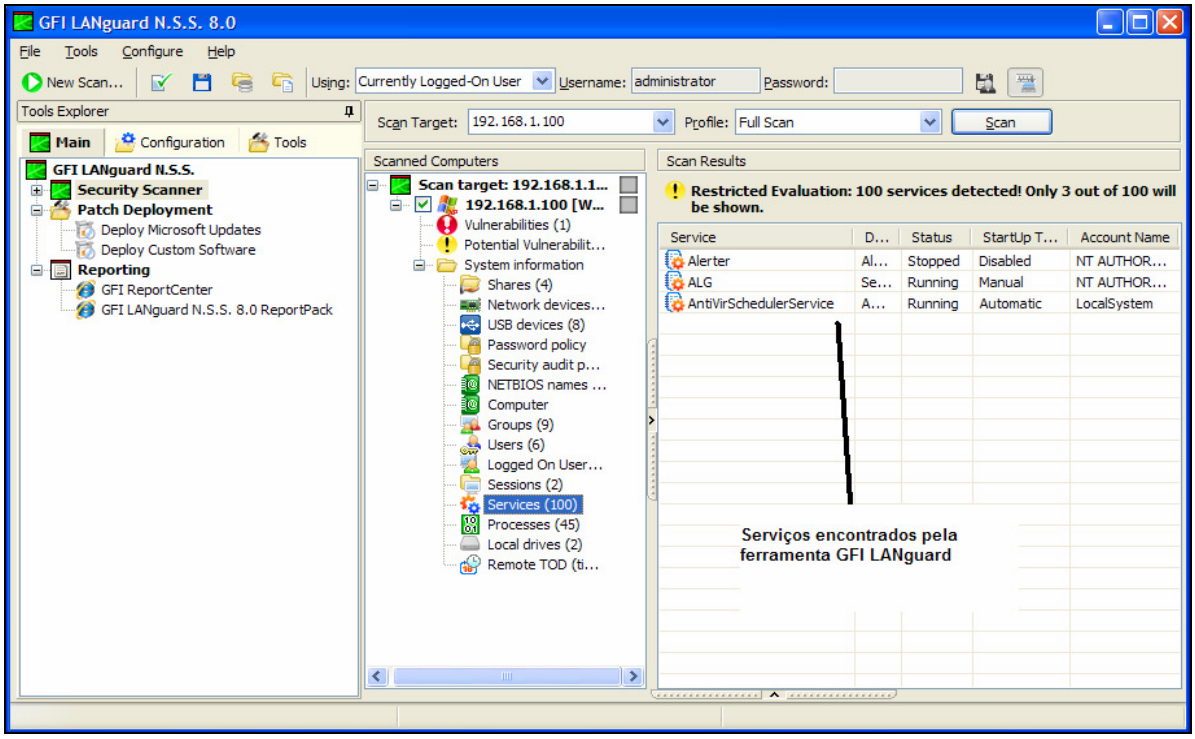

Figura 22. Serviços encontrados em execução
Fonte: Do autor.

Dessa forma, observou-se que a máquina virtual disponibiliza um encapsulamento para o IDS, de forma que não foi encontrada em nenhum momento nos testes qualquer evidência de que nesse *host* havia um sistema de detecção executando e analisando todo o tráfego de rede.

7.5 DIFICULDADES ENCONTRADAS

A principal dificuldade encontrada no desenvolvimento do projeto foi a falta de documentação, principalmente em relação à configuração do IDS Snort com o sistema operacional Windows. A fim de superar tais dificuldades foi utilizado como referência fóruns de discussão na Internet e monografias de outros acadêmicos.

Durante o processo de instalação e configuração das ferramentas ocorreram vários erros, onde muitos desses só foram resolvidos depois de dias de empenho e muito estudo.

A integração da ferramenta BASE com o banco de dados e a configuração do Snort foi o momento mais difícil do projeto, apesar dos muitos tutoriais na Internet, poucos deles são voltados para a plataforma Windows.

8 CONCLUSÃO

A segurança da informação muitas vezes não é tratada com a devida importância, o resultado disso pode ser observado nos jornais e noticiários, quando um ataque em massa ocorre devido há alguma vulnerabilidade de um determinado *software.* Constantemente estão sendo desenvolvidas ferramentas de proteção para combater esse tipo de situação. Porém muitas delas acabam sendo dribladas facilmente pelos invasores, isso faz com que se busque cada vez mais mecanismos para manter a integridade do sistema a ser protegido.

A instalação de uma ferramenta IDS é importante devido ao número de ataques que cresce constantemente, já que a mesma tem por característica detectar ataques ou invasão a um sistema. Mas foi observado que se o ataque for bem sucedido, poderia estar colocando em risco todas as informações que o IDS teria armazenado até então, pois o intruso após a invasão pode corromper essas informações.

Com a arquitetura proposta o IDS fica de forma camuflada dentro do ambiente virtual, onde fica protegido de ataques, pois não está visível para o resto da rede, mas está analisando todo o tráfego de forma despercebida. Foram realizados testes que comprovaram a invisibilidade do IDS, onde foi analisado o *host* real, e respectivamente os processos e serviços que estavam executando no momento e nada foi encontrado.

É importante salientar que as ferramentas e a arquitetura utilizadas e apresentadas neste trabalho não dispensam a utilização de *Firewall* e outras ferramentas como antivírus para a proteção da rede de computadores.

Como sugestão de trabalhos futuros pode-se ser implantar uma arquitetura maior utilizando máquinas virtuais, com outras ferramentas de máquina virtual, IDS e até mesmo com sistemas operacionais diferentes.

REFERÊNCIAS

ANÔNIMO. **Segurança Máxima:** o guia de um *harcker* para proteger se *site* na Internet e sua rede. 3. Ed. Traduzido por: Edson Furmankiewcz, Joana Figueiredo. Rio de Janeiro: Campus, 2000. 686 p.

BRITO, Leandro et al. **Virtualização de Sistemas Operacionais.** Disponível em: <http://thiagocavalcante.googlepages.com/Artigo_Virtualizacao.pdf>. Acesso em: 10 out. 2008.

CAMPELLO, Rafael Saldanha; WEBER, Raul Fernando. **Sistemas de Detecção de Intrusão.** In: LIVRO TEXTO DOS MINICURSOS. Florianópolis: UFSC, 2001. 43p

CASTRO, Arthur Bispo de ; SANTOS, Cleymone Ribeiro dos ; Paulo Lício de Geus / de GEUS, P. L. . MV6 - **Um mecanismo de transição baseado em máquinas virtuais**. In: IV Workshop em Segurança de Sistemas Computacionais, 2004, Gramado, RS. Anais do WSeg'04 (SBRC'04), 2004. p. 237-248.

CASTRO, Arthur Bispo de. **Máquinas Virtuais em Ambientes Seguros.** 2006. 73 f. Dissertação (Mestrado) - Universidade Estadual de Campinas, Campinas, 2006.

CARUSO, Carlos A. A.; STEFFEN, Flávio Deny. **Segurança em informática e de informações.** 2ª ed. Ver. E ampl. São Paulo: Senac, 1999.

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.

FERREIRA, Bárbara Chiavaro. **Sistema de Detecção de Intrusão.** Disponível em: <http://gravatai.ulbra.tche.br/~roland/tcc-gr/monografias/2003-2-tc2-Barbara_Chiavaro_Ferreira.pdf>. Acesso em: 10 out. 2008.

LAUREANO, Marcos. **Máquinas Virtuais E Emuladores:** conceitos, técnicas e aplicações. São Paulo: Novatec, 2006.

LAUREANO, Marcos; MAZIERO, Carlos; JAMHOUR, Edgard. **Detecção de Intrusão em Máquinas Virtuais.** 5º Simpósio de Segurança em Informática – SSI. ITA – São José dos Campos, 2003.

LAUREANO, Marcos; MAZIERO, Carlos; JAMHOUR, Edgard. **Detectando Intrusões na Máquina Virtual User-Mode Linux.** 5º Workshop sobre Software Livre - WLS. Sessão Paralela - Fórum Internacional Software Livre, 2004.

LAUREANO, M. A. P. **Uma abordagem para a proteção de detectores de intrusão baseada em máquinas virtuais**. Dissertação (Mestrado em Informática Aplicada) - Centro de Ciências Exatas e de Tecnologia, Pontifícia Universidade Católica do Paraná, Curitiba, 2004. 103 f.

NAKAMURA, Emílio Tissato; GEUS, Paulo Lício de. **Segurança de redes em ambientes cooperativos.** São Paulo: Berkeley Brasil, 2002. 320 p.

NORTHCUTT, Stephen et al. **Desvendando segurança em redes:** o guia definitivo para fortificação de perimetros de rede usando Firewalls, VPNs, roteadores e sistemas de detecção de invasores. Rio de Janeiro: Campus, 2002.

RAITZ, Luciano. **Utilização de Máquinas Virtuais para Implantar um Mecanismo Transparente de Detecção de Intrusão em Servidores Web**. Blumenau – SC: abril de 2005

SILBERSCHATZ, Abraham; GALVIN, Peter Baer; GAGNE, Greg. **Operating System Concepts**. John Wiley & Sons, Inc., New York, NY, USA, 6 edition, June 2001.

MIRANDA, Vitor de Deus. **Analise comparava entre as ferramentas VirtualBox, VMware Workstation e QEMU em ambiente Linux**. Lavras – MG, 2010.

APÊNDICE A – INSTALAÇÃO DO SISTEMA IDS SNORT USANDO APACHE, MYSQL E BASE

Neste apêndice é mostrado como foi instalado e configurado o IDS Snort, segue as ferramentas necessárias:

a) Apache Web Server:Para hospedar a consola BASE;

b) Snort: Para a captura do tráfego na rede;

c) WinPcap: É um 'device driver' que adiciona a capacidade para capturar e enviar dados em bruto da placa de rede;

d) MySQL Server: MySQL é um servidor de base de dados baseado em SQL;

e) ADODB: É uma biblioteca orientada a objetos escrita em PHP, que torna abstratas as operações de base de dados para obter portabilidade;

f) PHP: É uma linguagem de 'scripting' de uso geral amplamente utilizada;

g) Basic Analysis and Security Engine (BASE): é uma aplicação web para a visualização dos alertas do Snort IDS.

Após o download desses arquivos, é feita a instalação na seguinte ordem:

a) Instalar o Winpcap;

b) Instalar e configurar o Snort;

Após a instalação do Snort, tem-se que configurar o arquivo "snort.conf" conforme abaixo:

A variável 'home network' abaixo define a rede que deseja monitorizar:

Original: var HOME_NET any

Mudar para: var HOME_NET 192.168.1.0/24

A rede externa abaixo especifica uma ou mais redes de onde você acredita que os ataques serão originados.

Original: var EXTERNAL_NET any

Mudar para: var EXTERNAL_NET !$HOME_NET

Original: var RULE_PATH ../rules

Mudar para: var RULE_PATH c:\snort\rules

Original: # path to dynamic preprocessor libraries Acrescentar logo abaixo

dynamicpreprocessor file C:\Snort\lib\snort_dynamicpreprocessor\sf_dce2.dll

dynamicpreprocessor file C:\Snort\lib\snort_dynamicpreprocessor\sf_dcerpc.dll

dynamicpreprocessor file C:\Snort\lib\snort_dynamicpreprocessor\sf_dns.dll

dynamicpreprocessor file C:\Snort\lib\snort_dynamicpreprocessor\sf_ftptelnet.dll

dynamicpreprocessor file C:\Snort\lib\snort_dynamicpreprocessor\sf_sdf.dll

dynamicpreprocessor file C:\Snort\lib\snort_dynamicpreprocessor\sf_smtp.dll

dynamicpreprocessor file C:\Snort\lib\snort_dynamicpreprocessor\sf_ssh.dll

dynamicpreprocessor file C:\Snort\lib\snort_dynamicpreprocessor\sf_ssl.dll

Original:

preprocessor sfportscan: proto { all } \

memcap { 10000000 } \

sense_level { low }

Mudar para:

preprocessor sfportscan: proto { all } \

memcap { 10000000 } \

sense_level { low } \

logfile { portscan.log }

Nota: Abaixo de: '# output log_tcpdump: tcpdump.log' inserir a linha seguinte:

output alert_fast: alert.ids

Original: # output database: log, mysql, user=root password=test dbname=db host=localhost

Mudar para: output database: log, mysql, user=snort password=12345 dbname=snort host=localhost sensor_name=WinIDS

Original: include classification.config

Mudar para: include c:\snort\etc\classification.config

Original: include reference.config

Mudar para: include c:\snort\etc\reference.config

Original: # include threshold.conf

Mudar para: include c:\snort\etc\threshold.conf

Agora faça ‘save’ do ficheiro.

c) Instalar servidor Apache;

Na instalação é solicitado "Network Domain", "Server Name" e "Administrator's Email Address", pode-se usar, respectivamente: "localdomain", "localhost" e o seu endereço de e-mail. Deixe também seleccionada a opção "Run as a service for All users".

Também é necessário acrescentar no ficheiro de configuração 'httpd.conf' os seguintes comandos:

Original #LoadModule ssl_module modules/mod_ssl.so' e abaixo da mesma adicionar as três linhas seguintes:

LoadModule php5_module d:\win-ids\php\php5apache2_2.dll

AddType application/x-httpd-php .php

PHPIniDir d:\win-ids\php

Agora faça ‘save’ do ficheiro.

d) Instalar e configurar o PHP;

Após a instalação do PHP, abra ficheiro “php.ini”, procurar e alterar as variáveis abaixo:

Original: max_execution_time = 30

Mudar para: max_execution_time = 60

Original: ; session.save_path = "/tmp"

Mudar para: session.save_path = "c:\windows\temp"

(Verificar que a variável acima aponta para a diretoria correta WINDOWS\Temp)

Agora fazer ‘save’.

e) Testar a instalação do Apache e PHP;

Para isso criar um ficheiro ‘test.php’ com o conteúdo seguinte:

<? phpinfo(); ?>

Abrir um ‘browser’ e escrever 'http://localhost/test.php' se tudo correr bem aparecerão varias secções com informação sobre o PHP.

Verificar se informações seguintes estão corretas:

'Loaded Configuration File' é 'c:\php\php.ini'

'extension_dir' é 'c:\php\ext'

'include_path' é 'c:\php\pear'

'session.save_path' é 'c:\windows\temp'

f) Instalar e configurar o MySQL

Instalar Mysql, a seguir confirmar que está selecionada a opção 'Configure the MySQL Server Now' e click 'Finish'.

g) Criar a Base de Dados Snort

Abra uma janela MS-DOS e no ‘command prompt’ escreva: 'mysql -u root'

Agora estamos dentro da consola de administração MySQL

No ‘mysql prompt’ escreva 'create database snort;' e pressione Enter.

No ‘mysql prompt’ escreva 'create database archive;' e pressione Enter.

No ‘mysql prompt’ escreva 'show databases;' e pressione Enter.

Serão mostradas várias bases de dados.

Criar as Tabelas da Base de Dados Snort

No ‘mysql prompt’ escreva 'connect snort;' e pressione Enter.

No ‘mysql prompt’ escreva 'source d:\win-ids\snort\schemas\create_mysql' e pressione Enter.

No ‘mysql prompt’ escreva 'connect archive;' e pressione Enter.

No ‘mysql prompt’ escreva 'source d:\win-ids\snort\schemas\create_mysql' e pressione Enter.

Criar os Utilizadores e respectiva autenticação para Acesso às Bases de Dados

No ‘mysql prompt’ escreva 'set password for root@localhost = password('654321');' e pressione Enter.

(Agora para fazer ‘login’ na base de dados MySQL com o utilizador ‘root’ será necessária a ‘password’ ‘654321’.

No ‘mysql prompt’ escreva ‘\q’ ou 'quit;' para sair da base de dados.

No ‘command prompt’ escreva 'mysql -u root -p' e pressione Enter.

No ‘password prompt’ escreva ‘654321’ e pressione Enter.

No ‘mysql prompt’ escreva 'grant INSERT,SELECT,UPDATE on snort.* to snort identified by '12345';' e pressione Enter.

No ‘mysql prompt’ escreva 'grant INSERT,SELECT,UPDATE on snort.* to snort@localhost identified by '12345';' e pressione Enter.

No ‘mysql prompt’ escreva 'grant INSERT,SELECT,UPDATE,DELETE,CREATE on snort.* to base identified by 'xxx123';'

No ‘mysql prompt’ escreva 'grant INSERT,SELECT,UPDATE,DELETE,CREATE on snort.* to base@localhost identified by 'xxx123';' e pressione Enter.

No ‘mysql prompt’ escreva 'grant INSERT,SELECT,UPDATE,DELETE,CREATE on archive.* to base identified by 'xxx123';

No ‘mysql prompt’ escreva 'grant INSERT,SELECT,UPDATE,DELETE,CREATE on archive.* to base@localhost identified by 'xxx123';'

No ‘mysql prompt’ escreva ‘\q’

Faça ‘save’ e ‘reinicie’ o sistema.

h) Instalar ADODB

i) Instalar a Consola de Segurança BASE

Instale o BASE no diretório 'c:\apache\htdocs'

Abra uma janela MS-DOS e no ‘command prompt’ escreva:

'copy c:\apache\htdocs\base\base_conf.php.dist d:\win-ids\apache\htdocs\base\base_conf.php'

No ‘command prompt’ escreva 'mkdir c:\apache\htdocs\base\signatures'

No ‘command prompt’ escreva 'xcopy c:\snort\doc\signatures d:\win-ids\apache\htdocs\base\signatures /Q /Y'

Na diretoria 'c:\apache\htdocs\base' abra com o 'WordPad' o ficheiro 'base_conf.php'.

Procure e altere as variáveis abaixo:

Original: $BASE_urlpath = '';

Mudar para: $BASE_urlpath = 'http://winids/base';

Original: $DBlib_path = '';

Mudar para: $DBlib_path = 'd:\win-ids\adodb';

Original: $DBtype = '?????';

Mudar para: $DBtype = 'mysql';

Originais:

$alert_dbname = '?????';

$alert_host = '?????';

$alert_port = '?????';

$alert_user = '?????';

$alert_password = '?????';

Mudar para:

$alert_dbname = 'snort';

$alert_host = 'localhost';

$alert_port = '';

$alert_user = 'base';

$alert_password = 'xxx123;

Originais:

$archive_exists = 0; # Set this to 1 if you want access to the archive DB from

BASE

$archive_dbname = '?????';

$archive_host = '?????';

$archive_port = '?????';

$archive_user = '?????';

$archive_password = '?????';

Mudar para:

$archive_exists = 1; # Set this to 1 if you want access to the archive DB from BASE

$archive_dbname = 'archive';

$archive_host = 'localhost';

$archive_port = '';

$archive_user = 'base';

$archive_password = 'xxx123;

Original: $show_rows = 48;

Mudar para: $show_rows = 90;

Original: $show_expanded_query = 0;

Mudar para: $show_expanded_query = 1;

Original: $portscan_file = '';

Mudar para: $portscan_file = 'd:\win-ids\snort\log\portscan.log';

Original: $colored_alerts = 0;

Mudar para: $colored_alerts = 1;

Original: $priority_colors = array

('FF0000','FFFF00','FF9900','999999','FFFFFF','006600');

Mudar para: $priority_colors =

array('000000','FF0000','FF9900','FFFF00','999999');

Agora faça ‘save’.

j) Criar as Tabelas na Base de Dados para a consola BASE

Abra uma janela MS-DOS e no ‘command prompt’ escreva: 'mysql -u root -p'

Na ‘password’ escreva '654321'

No ‘mysql prompt’ escreva 'connect snort;

No ‘mysql prompt’ escreva 'source

c:\apache\htdocs\base\sql\create_base_tbls_mysql.sql'

No ‘mysql prompt’ escreva 'connect archive;

No ‘mysql prompt’ escreva 'source

c:\apache\htdocs\base\sql\create_base_tbls_mysql.sql'

No ‘mysql prompt’ escreva ‘\q’

Configuração Gráfica para BASE

Abra uma janela MS-DOS e no ‘command prompt’ escreva: 'cd c:\php'

No ‘command prompt’ escreva 'go-pear' < o:p>

No próximo ‘command prompt’ pressione ‘Enter’ para instalar 'System-Wide'

PEAR.

No próximo ‘command prompt’ pressione ‘Enter’ para continuar e retornar ao

‘command prompt’

Agora, No ‘command prompt’ escreva 'pear install Image_Color'

No ‘command prompt’ escreva 'pear install Log'

No ‘command prompt’ escreva 'pear install Numbers_Roman'

No ‘command prompt’ escreva 'pear install http://pear.php.net/get/Image_Canvas'

No ‘command prompt’ escreva 'pear install

http://pear.php.net/get/Numbers_Words-0.15.0'

No ‘command prompt’ escreva 'pear install

http://download.pear.php.net/package/Image_Graph-0.7.2.tgz'

Para testar a instalação PEAR,

No ‘command prompt’ escreva 'pear list'

Será mostrada uma lista pacotes PEAR

Criar a Segurança para a Consola BASE

Para acessar à consola BASE será necessário ‘username’ e ‘password’.

Abra uma janela MS-DOS e no ‘command prompt’ escreva: 'mkdir c:\apache\passwords'

No ‘command prompt’ escreva 'cd c:\apache\bin'

No ‘command prompt’ escreva 'htpasswd -c c:\apache\passwords\passwords base'

Nos próximos dois ‘prompt’ será necessário escrever a password e reescreve-la para o utilizador ‘base’

No ‘prompt’ 'New password:' escreva 'xxx123’

Em 'Re-type new password:’ escreva 'xxx123’

Na diretoria 'c:\apache\conf' abra o ficheiro 'httpd.conf' com o 'WordPad'.

Procure a secção de código abaixo:

<Directory />

Options FollowSymLinks

AllowOverride None

Order deny,allow

Deny from all

Satisfy all

</Directory>

E exatamente abaixo do referido código adicione as seguintes seis linhas:

<Directory "d:\win-ids\apache\htdocs\base">

AuthType Basic

AuthName "WinIDS"

AuthUserFile d:\win-ids\apache\passwords\passwords

Require user base

</Directory>

Procure e altere as linhas abaixo:

Original: DirectoryIndex index.html index.html.var

Mudar para: DirectoryIndex base_main.php

Original: Options Indexes FollowSymLinks

Mudar para: Options -Indexes FollowSymLinks

Faça ‘save’.

Agora faça ‘reboot’ ao sistema.

Depois do ‘reboot’ inicie um browser e escreva ' http://localhost/base'

Aparecerá uma caixa de dialogo de segurança e escreva no utilizador: ‘base, e na password: ‘xxx123’.

Está completa a instalação de um IDS

Apêndice B - Máquinas Virtuais Na Proteção de Sistemas de Detecção de Intrusão

William Pagnan[1], Paulo João Martins[2]

[1]Curso de Ciência da Computação - Unidade Acadêmica de Ciências, Engenharias e Tecnologias - Universidade do Extremo Sul Catarinense – UNESC

[2]Laboratório de Informática Aplicada - Curso de Ciência da Computação - Unidade Acadêmica de Ciências, Engenharias e Tecnologias - Universidade do Extremo Sul Catarinense – UNESC

pagnan@gmail.com, pjm@unesc.net

Abstract*. This article presents an architecture for use of intrusion detection by using virtual machines, each day that has spread further, with the main advantages of portability and cost. The proposed architecture makes use of virtual machines to stop the intrusion detection system encapsulated making it invisible and inaccessible. The tests presented show the importance of using virtual machines, and the ability of isolation that they allow, can be used to encapsulate intrusion detection systems.*

Resumo. *Este artigo apresenta uma arquitetura para uso de detectores de intrusão através da utilização de máquinas virtuais, que a cada dia vem se disseminando mais, com as principais vantagens de portabilidade e custo. A arquitetura proposta faz uso de máquinas virtuais para deixar o sistema de detecção de intrusão encapsulado tornando-o invisível e inacessível. Os testes apresentados, mostram a importância do uso de máquinas virtuais, e a capacidade de isolamento que as mesmas permitem, podendo ser usadas para encapsular sistemas de detecção de intrusão.*

h) Introdução

A utilização de ambientes computacionais tem se difundido de forma crescente, disponibilizando informações importantes para o dia-a-dia das pessoas e das organizações. Porém, com a disseminação destes ambientes aumenta a preocupação no que se refere à segurança dos dados e informações, pois estas não devem ser acessadas por pessoas não autorizadas. Existem pessoas (*crackers*) que visam acessar sistemas computacionais a fim de alterar, apagar dados, entre outros.

Assim, a fim de proporcionar segurança nesses sistemas, bem como aos seus usuários, existem diferentes alternativas que objetivam proteger os dados, como por exemplo, *firewall,* virtualização de sistemas operacionais e sistemas de detecção de intrusão (*Intrusion Detection System* – IDS). É crescente o uso de sistemas virtualizados, já que os mesmos podem ser usados para ampliar o nível de segurança de um ambiente computacional contra ataques ou intrusões indesejadas (CHEN, 2001).

Além disso, pode-se usufruir dos IDS que monitoram continuamente as atividades de um ambiente computacional, visando encontrar evidências de intrusão (LAUREANO, 2004).

i) Máquinas Virtuais

Uma máquina real é composta por vários componentes físicos que permitem que as operações de um sistema operacional e suas aplicações se realizem. Já uma máquina virtual ou Virtual Machine (VM) pode ser definida como uma cópia idêntica isolada de uma máquina real (CHEN, 2001). Com o enorme crescimento do poder de processamento dos computadores hoje em dia, a utilização de máquinas virtuais vem sendo uma ótima opção para sistemas da computação, pelos seus custos e portabilidade, inclusive para sistemas de segurança (LAUREANO, 2006), que é um dos objetivos deste trabalho de conclusão.

Cada máquina virtual é completamente isolada uma das outras, sendo assim, sua utilização proporciona um aumento de segurança para os recursos do sistema, pois os mesmos estão de certa forma protegidos. As aplicações não confiáveis podem ser executadas em máquinas virtuais separadas, isolando assim o perigo de danos na máquina real (CASTRO; SANTOS; GEUS, 2004).

As máquinas virtuais proporcionam um isolamento entre o sistema operacional convidado e sistema operacional hospedeiro, sendo assim um ataque ao obter sucesso na invasão da aplicação e dominar o sistema convidado, tem ainda que comprometer o *software* da máquina virtual para depois poder atingir a máquina real (CASTRO; SANTOS; GEUS, 2004).

Os sistemas operacionais e aplicações são desenvolvidos para aproveitar ao máximo dos recursos do *hardware*, porém ao longo dos anos muitas plataformas operacionais diferentes foram criadas, causando incompatibilidades entre si. O uso de máquinas virtuais permite compatibilizar diferentes plataformas, pois ela ao ser instalada na máquina real cria uma "camada" que é chamada de virtualização.

É um recurso de software que possibilita a execução virtual de um ou mais sistemas operacionais ao mesmo tempo em apenas uma máquina física, que já contém um sistema operacional nativo instalado. Primeiramente é instalado um software no sistema operacional da máquina real, e a partir deste, outros sistemas operacionais podem ser adicionados de forma virtual (BRITO et al., 2008).

Atualmente este tem chamado muita a atenção de pequenas e grandes empresas, pois a utilização desse recurso pode trazer diversos benefícios, como economia de energia elétrica e de espaço. Sendo assim as empresas podem eliminar custos sem trazerem impactos a funcionalidades de seus equipamentos (BRITO et al., 2008).

Segundo Laureano (2006) uma máquina virtual é um ambiente criado por um monitor de máquinas virtuais ou *Virtual Machine Monitor* (VMM) também conhecido como hypervisor, que é responsável por simular os recursos de *hardware* para a utilização das máquinas virtuais e traduzir suas requisições para a máquina real. O VMM fornece uma interface muito perfeita do *hardware*, de maneira que as aplicações e sistemas que estão em cada VM "pensam" estar trabalhando em uma máquina real.

j) Técnicas de Virtualização

As técnicas mais utilizadas atualmente para a virtualização são a paravirtualização, virtualização total e recompilação dinâmica.

Na paravirtualização é necessário fazer modificações no sistema virtualizado (sistema convidado), para que o mesmo consiga acessar recursos do *hardware* diretamente. O acesso é monitorado pelo monitor de máquinas virtuais, que repassa ao sistema todas as informações necessárias para o acesso. A paravirtualização é umas das técnicas que tem a melhor *performance*, pois ela reduz a complexidade das máquinas virtuais, sendo assim

compensa as modificações que terão que ser feitas nos sistemas convidados (LAUREANO, 2006).

Já na virtualização total o sistema não sofre qualquer alteração, sendo este o principal benefício desta técnica, porém esta técnica torna-o mais lento e precisa da intervenção do monitor de máquinas virtuais para o gerenciamento do acesso a memória, ao disco e para implementar alternativas para que operações privilegiadas possam ser executadas em certos processadores que não suportam a virtualização nativa (LAUREANO, 2006).

A recompilação dinâmica também é muito utilizada, essa técnica faz a compilação de partes do código durante a execução do sistema, podendo assim ajustar o código gerado, produzindo um ambiente idêntico ao original para que o programa consiga informações que não estão disponíveis em um compilador estático tradicional. Ela também pode ser aplicada como parte de uma estratégia de otimização adaptável para executar uma representação portátil do programa (LAUREANO, 2006).

k) Vantagens e Desvantagens de Máquinas Virtuais

As máquinas virtuais trazem diversas vantagens para os sistemas de computação, sendo as principais (RAITZ, 2005):

1. executar diferentes sistemas operacionais sobre o mesmo *hardware*, ao mesmo tempo;
2. facilitar testes de novos sistemas operacionais;
3. simular configurações e situações diferentes do mundo real;
4. garantir a portabilidade de aplicações;
5. diminuição de custos de *hardware*;
6. facilidade na migração, gerenciamento e replicação de aplicações ou sistemas operacionais;
7. simular alterações e falhas no *hardware* para testes ou reconfigurações de sistemas operacionais, aumentando a confiabilidade e escalabilidade dessas aplicações.

As máquinas virtuais também trazem com si algumas dificuldades, sendo as principais encontradas as seguintes (LAUREANO, 2006):

f) custo do processo de virtualização;
g) processador não virtualizado;
h) diversidade de equipamentos, sendo assim, o trabalho do monitor de máquinas virtuais é sobrecarregado;
i) preexistência de softwares já instalados nos PCs ou *desktops*, onde a máquina virtual tem que ser disponibilizada, mas estas não podem afetar os sistemas já existentes na máquina;
j) custo adicional de execução dos processos em comparação á de uma máquina real.

Principais categorias de máquinas virtuais (CASTRO, 2006): emuladores; VMM do tipo I; VMM do tipo II; paravirtualização; virtualização de linguagem de alto nível e no nível do sistema operacional.

3. IDS – Intrusion Detection System

Nos últimos anos a tecnologia de detecção de intrusão (*Intrusion Detection System* - IDS) tem se mostrado uma grande aliada na área da segurança, como fonte de estudos para os administradores da mesma (LAUREANO, 2006).

O termo intrusão pode ser caracterizado como uma violação da política de um sistema. Este tipo de sistema monitora e analisa os eventos de uma rede de computadores, visando encontrar qualquer atividade que comprometa a confiabilidade, integridade e disponibilidade de recursos computacionais ou de rede (NORTHCUTT et al., 2002). Existem dois tipos de IDS: sistemas de detecção de intrusão baseados em rede (*Network-Based Intrusion Detection* – NIDS), e sistemas de detecção de intrusão baseados em *host* (*Host-Based Intrusion Detection* (HIDS).

3.1. Exemplos de Sistemas de Detecção de Intrusão

A área de detecção de intrusão é recente, pois a cada dia são descobertas novas falhas e vulnerabilidades nos sistemas. Exemplos de alguns sistemas que tentam contornar essa situação: Snort, RealSecure, Intruder Alert e Nuzzler Basic.

4. Implantação de Um Sistema IDS em uma Máquina Virtual

A arquitetura aqui desenvolvida, demonstrar uma maneira segura para proteger o IDS através do uso de máquina virtual. O sistema operacional a ser protegido juntamente com seus *softwares* deve ser executado no ambiente virtual com sua configuração padrão como se estivesse em um *Hardware* comum.

Para que se possa desenvolver a proposta, foi utilizada a seguinte arquitetura:

a) Sistema operacional Windows XP Professional (Service Pack 3);
b) IDS Snort (Versão2_8_6_1);
d) MySql (Versão 5.1.50);
e) Apache (Versão 2.2.16);
f) Basic Analysis and Security Engine (BASE) (Versão 1.4.5);
g) Máquina Virtual VirtualBox (Versão 3.2.8).

Tanto no sistema *host* como no sistema anfitrião foi instalado o sistema Windows XP Profissional. No VirtualBox, além do sistema convidado que será utilizado para mostrar os testes, também existe outra máquina virtual, com o sistema Linux Ubuntu 9.

O ambiente virtual foi criado usando o método de utilização de discos virtuais, pois assim não se tem a necessidade de ter que criar partições no disco rígido, facilitando a instalação do mesmo e até sua cópia.

A configuração de rede deve ser modificada para que a arquitetura proposta tenha o funcionamento correto. De maneira que o sistema virtualizado não tenha protocolos de rede ativos, tornando-o inacessível e protegido de qualquer ataque via rede. Porém, é necessário criar uma conexão direta a rede local, sendo que essa alteração é feita através do VirtualBox antes da inicialização do mesmo, para que assim o IDS instalado no sistema convidado tenha acesso aos dados que trafegam no ambiente não virtualizado, pois por padrão uma máquina virtual cria sua própria rede, e torna impossível a análise das informações direcionadas a rede local.

Após a montagem da arquitetura, foi instalado o IDS Snort no sistema convidado, sendo configurado para capturar todos os pacotes que tenham como destino ou origem *hosts* da rede em análise. Para que isso aconteça é necessário que sejam feitas alterações no arquivo "snort.conf", arquivo esse que contém a configuração do Snort.

Posteriormente o MySql foi instalado para que os pacotes capturados fossem gravados no banco de dados, e assim disponibilizando-os para futuras consultas para análise dessas informações. Em seguida o Apache foi instalado, permitindo assim o uso da

ferramenta BASE, pois a mesma necessita estar instalada em um servidor *Web* para funcionar. O BASE foi a ferramenta utilizada para a consulta dos pacotes capturados.

5. Realização de Testes na Máquina Virtual

Foram utilizadas as ferramentas Nmap e GFI LANguard. O Nmap é *software* livre utilizado para realizar uma varredura de portas e serviços, sendo muito eficiente para avaliar a segurança de um computador, buscando detectar quais serviços estão rodando no mesmo. O GFI LANguard também é um scanner, que procura falhas de segurança emitindo um relatório ao final informando quais pontos estão com falhas.

A Figura 1 nos mostra a leitura do Snort após uma varredura pelos software sniffers, pode-se observar uma grande quantidade de alertas encontrados, a categoria em que eles se encontram (TCP , UDP e ICMP), de qual IP foram enviados e qual o destino. Todos os alertas aqui ilustrados estão armazenados em um banco de dados, provendo assim a possibilidade de futuramente o administrador consulta-los e analisar-los.

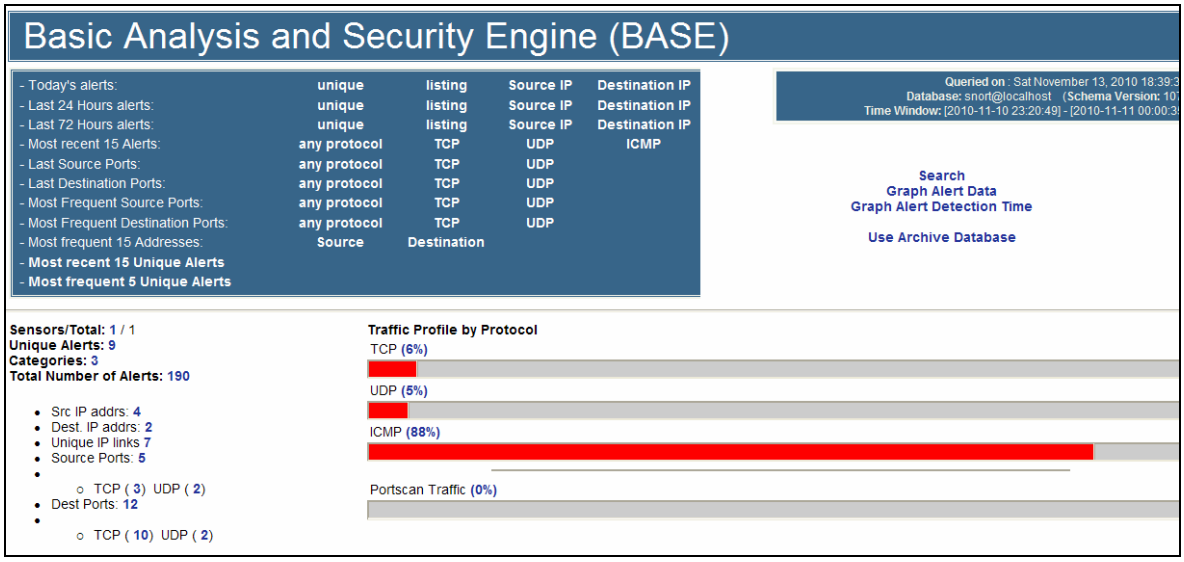

Figura 1. Resultados após varredura de portas – interface principal
Fonte: Do autor

6. Realização de Testes no Sistema Hospedeiro

Após ter sido feito testes com o sistema IDS dentro da máquina virtual e o mesmo obtendo sucesso na análise de todo o tráfego de rede, foram feitos agora testes no sistema hospedeiro (ambiente real) para tentar encontrar algum rastro de que o mesmo tivesse um ambiente virtual ou sistema de detecção de intrusão em execução.

Primeiro foi feito uma varredura de ataques com a ferramenta Nmap, onde esta encontrou apenas serviços de rede do Windows e um banco de dados executando no momento. Foram efetuadas diversas varreduras e nenhuma delas apresentou informação sobre IDS ou máquina virtual sendo executada, sendo que no momento do *scan* o *host* em análise estava com o ambiente virtual em plena execução e o IDS capturando informações da rede.

Depois foi executada a busca com a ferramenta GFI LANguard, onde também não encontrou nada além de serviços como anti-vírus e alguns processos de Internet do Windows.

Dessa forma, observou-se que a máquina virtual disponibiliza um encapsulamento para o IDS, de forma que não foi encontrada em nenhum momento nos testes qualquer

evidência de que nesse *host* havia um sistema de detecção executando e analisando todo o tráfego de rede.

Referência

BRITO, Leandro et al. **Virtualização de Sistemas Operacionais.** Disponível em: <http://thiagocavalcante.googlepages.com/Artigo_Virtualizacao.pdf>. Acesso em: 10 out. 2008.

CAMPELLO, Rafael Saldanha; WEBER, Raul Fernando. **Sistemas de Detecção de Intrusão.** In: LIVRO TEXTO DOS MINICURSOS. Florianópolis: UFSC, 2001. 43p

CASTRO, Arthur Bispo de ; SANTOS, Cleymone Ribeiro dos ; Paulo Lício de Geus / de GEUS, P. L. . MV6 - **Um mecanismo de transição baseado em máquinas virtuais**. In: IV Workshop em Segurança de Sistemas Computacionais, 2004, Gramado, RS. Anais do WSeg'04 (SBRC'04), 2004. p. 237-248.

CASTRO, Arthur Bispo de. **Máquinas Virtuais em Ambientes Seguros.** 2006. 73 f. Dissertação (Mestrado) - Universidade Estadual de Campinas, Campinas, 2006.

CARUSO, Carlos A. A.; STEFFEN, Flávio Deny. **Segurança em informática e de informações.** 2ª ed. Ver. E ampl. São Paulo: Senac, 1999.

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.

FERREIRA, Bárbara Chiavaro. **Sistema de Detecção de Intrusão.** Disponível em: <http://gravatai.ulbra.tche.br/~roland/tcc-gr/monografias/2003-2-tc2-Barbara_Chiavaro_Ferreira.pdf>. Acesso em: 10 out. 2008.

LAUREANO, Marcos. **Máquinas Virtuais E Emuladores:** conceitos, técnicas e aplicações. São Paulo: Novatec, 2006.

LAUREANO, M. A. P. **Uma abordagem para a proteção de detectores de intrusão baseada em máquinas virtuais**. Dissertação (Mestrado em Informática Aplicada) - Centro de Ciências Exatas e de Tecnologia, Pontifícia Universidade Católica do Paraná, Curitiba, 2004. 103 f.

NAKAMURA, Emílio Tissato; GEUS, Paulo Lício de. **Segurança de redes em ambientes cooperativos.** São Paulo: Berkeley Brasil, 2002. 320 p.

NORTHCUTT, Stephen et al. **Desvendando segurança em redes:** o guia definitivo para fortificação de perimetros de rede usando Firewalls, VPNs, roteadores e sistemas de detecção de invasores. Rio de Janeiro: Campus, 2002.

RAITZ, Luciano. **Utilização de Máquinas Virtuais para Implantar um Mecanismo Transparente de Detecção de Intrusão em Servidores Web**. Blumenau – SC: abril de 2005

SILBERSCHATZ, Abraham; GALVIN, Peter Baer; GAGNE, Greg. **Operating System Concepts**. John Wiley & Sons, Inc., New York, NY, USA, 6 edition, June 2001.

MIRANDA, Vitor de Deus. **Analise comparava entre as ferramentas VirtualBox, VMware Workstation e QEMU em ambiente Linux**. Lavras – MG, 2010.