1. Wirtualizator stanowi warstwę pośrednią pomiędzy producentem zasobów (np. pamięcią RAM) a konsumentem (np. systemem operacyjnym). Z punktu widzenia konsumenta jest tylko jeden konsument, a z punktu widzenia konsumenta jest tylko jeden producent zasobu. W rzeczywistości – dzięki maszynie wirtualnej – może ich być wiele. Maszyna wirtualna jest (zwykle) dla nich transparentna.

1. System gospodarza to ten, na którym jest uruchomiona maszyna wirtualna.
2. System operacyjny gościa to ten, który znajduje się wewnątrz maszyny.
3. Hypervisor – czyli wirtualizator albo monitor maszyny wirtualnej – to "hipernadzorca": ten, który stoi nad supervisorem (OSem).
4. Wirtualizator natywny jest uruchamiany bezpośrednio na sprzęcie, a nie w systemie gospodarza. Przykład – Xen.
5. Większość wirtualizatorów jest gościnna – uruchamiana w systemie gospodarza.

1. System gościa musi działać tak, jak uruchomiony na sprzęcie.
2. Hypervisor przydziela i kontroluje zasoby systemowi gościa i jest w stanie zapewnić, że gość nie wykorzysta żadnych dodatkowych.
3. Nie wymaga się 100% wydajności systemu niewirtualizowanego, ale jakiejś „rozsądnej”.
4. W praktyce żadne z tych kryteriów nie jest w 100% spełnione.

1. Z kryterium kontroli zasobów wynika pośrednio, że hypervisor potrafi zapisać stan maszyny – wszystkie wykorzystywane zasoby, wykonywane procesy etc – jako plik obrazu (snapshotu, checkpointu).
2. W przypadku awarii zwykle szybciej jest wczytać taki obraz niż przeprowadzić restart.
3. W wielu firmach ze względów bezpieczeństwa oddziela się serwery pocztowe, DHCP etc na oddzielne maszyny – mimo, że nie potrzebują one 100% ich mocy. Użycie wirtualizacji jest tańsze.
4. Honey-pot to maszyna-przynęta służąca do łapania nowych wirusów itp.

- Istnieją instrukcje, które pozwalają na odczytanie rejestrów systemowych w trybie użytkownika
- Nie ma sprzętowych mechanizmów wykrywania wszystkich instrukcji, które nie działają poprawnie w wirtualnym środowisku
- Należy tak przerywać wykonanie kodu (tworzyć breakpointy), żeby nadzorca mógł odpowiednio reagować
- Modyfikacje jakich użyjemy nie mogą zostać wykryte przez gościa

1. (ad.1) Na szczęście nie ma tych instrukcji tak wiele. Niektóre z nich to: LAR, LSL, VERR, VERW, SGDT, SIDT, SLDT, SMSW, STR. Więcej szczegółów w artykule pod adresem http://www.floobydust.com/virtualization/lawton_1999.txt
2. (ad.4) W przeciwnym przypadku mogłoby dojść do zmiany sposobu w jaki wykona się program gościa
3. (ad.4) W praktyce nadzorca musi działać jak nieintruzyjny programowy debugger systemu gościa

- Należy tak ustawić breakpointy, żeby nigdy nie dopuścić do wykonania kodu, który nie był jeszcze sprawdzony
- Trzeba uważać na kod, który zagląda do już przeskanowanej części programu
- Jeszcze większym problemem jest kod, który modyfikuje przeskanowaną część
- Istnieją sztuczki, dzięki którym tworzy się stronę z kodem wykonywalnym, którego nie można odczytać i zapisać

1. (ad.2) Należy w takim przypadku ukryć nasze modyfikacje. Można tego dokonać poprzez stworzenie niezmodyfikowanej strony w pamięci, do której będą się odnosiły odczyty
2. (ad.3) Dość skutecznym obejściem problemu jest umieszczenie przeskanowanego kodu w pamięci, na stronie z flagą zabezpieczającą przed zapisem. Wtedy każda próba modyfikacji takiej strony wywoła przerwanie i nadzorca będzie mógł zareagować
3. (ad.4) Sztuczki bazują na tym, że w większości procesorów cache tablic TLB dotyczących instrukcji i danych są odseparowane
4. Naturalnie problemów z wirtualizacją programową na tej architekturze jest znacznie więcej. Można o nich przeczytać w artykule, do którego link podany jest wyżej

Intel Virtualization Technology (IVT)
- zwana też Vanderpool
- Rozszerzenie architektury x86 o wirtualizację
- VT-x - dla 32-bitowych procesorów Intela
- VT-i - dla procesorów IA-64 (Itanium)

AMD virtualization (AMD-V)
- nazwa wewnętrzna Pacifica
- dla 64-bitowych procesorów x86
- stworzone równolegle z rozwiązaniem Intela

1. Oba rozwiązania powstały w latach 2005-2006. Nie są ze sobą kompatybilne, ale spełniają te same funkcje
2. Rozwiązują problem wirtualizacji poprzez umożliwienie maszynie wirtualnej na uruchomienie hypervisora. Dzięki temu można uruchomić niezmodyfikowany system operacyjny gościa
3. Na ten moment technologia ta jest uznawana za niedojrzałą. Niektóre badania dowodzą, że jest wolniejsza od wirtualizacji programowej. Zapewne w najbliższych latach się to zmieni

- Hierarchiczne domeny – Ringi są zorganizowane w hierarchię.
- Sprzętowe wsparcie dla SO.
- Tradycyjny podział:
  - 0 – kernel
  - 1, 2 – sterowniki urządzeń, wirtualizacja
  - 3 – procesy użytkownika
- Ring -1 jest jednym ze sposobów wirtualizacji SO o zamkniętych źródłach (AMD-V, Intel VT)

1. Ring – to w pewnym sensie tryb pracy procesora determinujący poziom uprawnień wykonywanego kodu.
2. Ring 0 - najbardziej uprzywilejowany
3. Przejścia pomiędzy ringami są realizowane np. przez instrukcje (INT), przez przerwania systemowe.
4. Każdy Ring „zawiera” ten o mniejszym numerze, tzn. w naturalny sposób może wykonywać w nim kod wymagający mniejszych praw niż dostępne.
5. Hypervisor może korzystać z Ring -1, a Gość w Ring 0, co pozwala go nie modyfikować, wymaga to jednak odpowiedniego procesora.
6. Na podstawie: http://en.wikipedia.org/wiki/Ring_(computer_security)

1. Technika wirtualizacji pełnej stara się uruchamiać możliwie dużo kodu gościa bezpośrednio na procesorze. Musi jedynie identyfikować instrukcje niebezpieczne oraz (zwykle) funkcje jądra i je emulować. Wiąże się to ze znacznym wzrostem efektywności w stosunku do emulacji całego sprzętu.
2. Wzrost wydajności ma swoją cenę - systemy gościa i gospodarza muszą pracować na tej samej architekturze (z oczywistych względów).

Microsoft Virtual PC
Przedstawienie

- Oryginalny producent - Connectix
  - Pierwsze wydanie - czerwiec 1997
  - Początkowo emulator x86 dla Mac-ów na PowerPC
  - Pierwsza wersja na MS Windows: czerwiec 2001
- Przejęcie przez Microsoft w lutym 2003
- Platforma docelowa: MS Windows
  - ...oraz Mac OS na PowerPC
- Darmowy od lipca 2006

1. Virtual PC dla Mac OS jest emulatorem x86 na procesory PowerPC. 08.06 MS ogłosił, że nie będzie wypuszczał wersji dla procesorów x86; Macow na PowerPC sie juz nie produkuje, więc ten segment rynku jest przejęty prawie całkowicie przez nowocześniejszy Parallels Desktop.

- Wersja dla x86
  - Zasada działania - pełna wirtualizacja
    - Wersja 2007 - wykorzystuje AMD-V i IVT
  - Wsparcie dla sieci, napędów CD/DVD, dźwięku, drukarek...
  - Virtual Machine Additions - dla systemów Microsoftu
    - M.in. zwiększenie wydajności, wsparcie dla akceleracji 3D.
  - Zapisywanie obrazu maszyny
  - Wydajność - prawie natywna
- Wersja dla PowerPC
  - Emulator x86 dla Mac OS X
  - Bardzo niska wydajność
  - Praktycznie zażegnany

1. VMA to sterowniki instalowane w systemie gościa znacząco wpływające na polepszenie wydajności i funkcjonalności.

VMWare Workstation
Przedstawienie

- Producent: VMWare
  - VMWare Server
  - VMWare Fusion
- Pierwsze wydanie - 1999
- Platformy docelowe: Windows i Linux
- 30-dniowy darmowy okres próbny
- VMWare Player - darmowy
  - Nie tworzy obrazów, jedynie uruchamia
  - Można je ściągnąć z Internetu
  - Trochę gorsza wydajność, brak SMP

1. VMWare Workstation jest najbardziej rozbudowanym produktem w tej rodzinie. VMWare Server (z odrobinę mniejszymi możliwościami, bardziej pod kątem zastosowań serwerowych) jest dostępny całkowicie za darmo. VMWare Fusion to wersja dla Mac OS.

- Zasadniczo platforma wirtualizacyjna
  - Od wersji 6.0 - eksperymentalne wsparcie dla parawirtualizacji
  - Otwarty protokół VMI, włączony do jądra 2.6.21
- Rozbudowana funkcjonalność
  - Praktycznie wszystkie urządzenia peryferyjne
  - Zarządzanie snapshotami
  - Rozbudowane wsparcie dla sieci
  - Ułatwione administrowanie
  - Szyfrowanie
- Wydajność - prawie natywna

1. Protokół VMI – Virtual Machine Interface – został opracowany jako otwarty projekt wspólnie ze środowiskiem Linuksowym.

- Produkcja i testowanie oprogramowania
- Demonstracje produktów u klienta
- Również dydaktyka

1. Marketerzy VMWare twierdzą, że możliwość pokazania gotowego i w pełni działającego systemu „u klienta” zwiększa szansę sprzedania go o X%... :)

- Zamiast emulować sprzęt, lepiej dostarczyć API – hypercall
- ...najlepiej mieć wsparcie w procesorze ;)

1. Gość, zamiast wykonywać syscalle, wykonuje hypercalle obsługiwane przez Hosta.
2. Pierwszym ślad tej techniki to mechanizm DIAG w IBM'owskich mainframe-ach z lat '60. Twórcom przyświecała chęć efektywniejszego wykorzystania czasu procesora przez wielu użytkowników.
3. Przy pomocy rozwiązań sprzętowych (AMD-V, Intel VT) można bardzo łatwo przekazywać sterowanie do hypervisora, bez konieczności np. dodatkowego śledzenia wirtualizowanego kodu.

Parallels Desktop for Mac

- Wymaga MacOS X, wspiera w pełni Intel VT
- VirtualPC for Mac – protoplasta, obecnie porzucony
- Zaawansowane współdzielenie plików pomiędzy Hostem a Gościem
- Snapshoty
- DirectX z Wine
- Oficjalnie wspierane systemy Gościa:
  - Windows 3.1 - Vista
  - Linux (SUSE, Red Hat, Debian, Fedora Core, Mandriva, Xandros)
  - FreeBSD, OpenBSD 3.8
  - OS/2, eComStation
  - Solaris
  - MS-DOS

1. PD jest w zasadzie przeniesieniem koncepcji emulatora VPC do obecnej sytuacji w świecie Maca, zapewnia użytkownikom Maców dostęp do oprogramowania (w tym gier Windowsowych). Dzięki temu, że Apple wyposaża swoje komputery w Intelowskie Core zniknęła potrzeba emulacji szerszego zestawu instrukcji (x86 na PPC)
2. Można bardzo wygodnie przenosić pliki pomiędzy systemami, podczas działania VM.
3. Można zapisywać stan VM i potem go wznowić, np. gdy potrzebujemy mocy Maca, a nie możemy przerwać działania programu na VM.
4. Parallels Inc. wykorzystało implementację DirectX z Wine, żeby mieć obsługę i akceleracją 3D na emulowanej maszynie, ale nie opublikowało od razu zmienionych kodów źródłowych, co narusza GPL (Wine). Po protestach ze strony środowiska GNU po ponad tygodniu pojawiły się zmodyfikowane źródła.

XEN

- Ian Pratt, University of Cambridge, 2003
- Wspiera IA-32, x86-64, IA-64 oraz PowerPC
- Host – Linux lub NetBSD (zmodyfikowane)
- Gość – różne systemy unixowe (modyfikowane); Windows (od Xen 3.0)
- Wolny (free), ale istnieje komercyjna implementacja: Citrix XenServer Enterprise Edition
- Novell SUSE 10, Red Hat's RHEL 5/Fedora 7, Sun Microsystems Solaris 10, Debian Etch, Ubuntu 6.10

1. Citrix to komercyjna implementacja Xen, kosztuje od ok. $1600; jest rozprowadzana przez Della oraz HP wraz z ich serwerami.
2. Wymienione dystrybucje Linuksa oraz Solaris zawierają Xen w standardzie; do innych można go względnie łatwo zainstalować.
3. W Xenie występuje pojęcie *domeny* – jest to w uproszczeniu wirtualna maszyna, na której działa OS, przy czym może być tylko jedna domena dom0 – hypervisor oraz wiele zarządzanych przez niego domen Gościa - domU.
4. Gdy procesor nie wspiera wirtualizacji – dom0 działa w ringu 0, a domU – w ringu 1.
5. Gdy mamy wsparcie w procesorze – Xen dom0 działa w ringach 0-3 i korzysta z prawdziwego sprzętu, domU działają w ringach 0-3, ale jej (domU) „sprzęt" jest tylko zasobem dom0.

XEN
wspierane systemy Gościa

- Linux 2.6.23+, patche dla wcześniejszych
- Minix
- Plan 9 from Bell Labs
- NetBSD 2.0+
- OpenBSD
- FreeBSD
- OpenSolaris
- NetWare
- GNU/Hurd/Mach (gnumach-1-branch-Xen-branch)
- OZONE (Xen v1.2)
- oraz Windows XP :)

1. Linux od wersji 2.6.23 zawiera wsparcie dla Xen włączone do głównego drzewa, wcześniejsze trzeba łatać.
2. Uruchomienie WXP wymaga wsparcia w sprzęcie oraz Xen 3.0; ciekawe, czy ktoś podjąłby się przerobienia W2k (źródła wyciekły swego czasu...).
3. Powstał port WXP na Xen w ramach badań naukowych, ale licencja zabrania jego rozpowszechniania.
4. Windows 2008 ma zawierać pełne wsparcie dla Xena, ale te inforamcje nie są do końca oficjalne, poza tym MS nie zawsze dotrzymuje danego słowa.

Zalety:
- Szybkość
- Przemieszczanie domeny „w locie” (60-300 ms)

Wady:
- Konieczność modyfikowania jądra gościa lub posiadania odpowiedniego procesora
- Skomplikowany system
- Dość trudna instalacja (w porównaniu do UML czy VirtualPC)

1. Xen jest bardzo szybki, zwłaszcza na tle innych rozwiązań tej klasy, nieznacznie wolniejszy niż natywny system; zdobywa dzięki temu popularność.
2. Działającą domenę Xena można przenieść na inny komputer w sieci LAN, RAM jest przenoszony iteracyjnie podczas działania. Gdy wszystko jest skopiowane potrzeba 60-300 ms na przełączenie wykonania na drugą maszynę; z zewnątrz wygląda to jakby nie było żadnej przerwy w działaniu np. serwera bazy danych.
3. Funkcjonalność i szybkość Xena sprawia, że jest stosunkowo trudny i poznanie go zajmuje dość dużo czasu.
4. Inaczej niż w przypadku UML czy VPC trzeba ściągnąć więcej plików, poznać organizację, sposób tworzenia i zarządzania domenami (choć np. w Suse graficzny konfigurator Yast za to odpowiada, nie jest więc tak źle, sama instalacja jest też bezproblemowa na Suse), spachować Hosta, system gościa, kompilować jądra ze źródeł...

- Szybsze niż inne rodzaje wirtualizacji
- Wiele instancji tego samego systemu operacyjnego
- Wszystkie instancje pracują na tym samym jądrze
- Wygodne do testowania nowego oprogramowania
- Zwiększa bezpieczeństwo pracy dzięki odizolowaniu różnych części systemu

1. (ad.3) Jedynie jądro systemu jest wszędzie takie samo. Biblioteki, oprogramowanie etc. mogą być inne w każdym VE
2. (ad.3) W szczególności w przypadku Linuksa możliwe jest uruchomienie różnych dystrybucji, oczywiście na tym samym jądrze
3. (ad.4) Można stworzyć obraz VE, zainstalować nowe oprogramowanie i sprawdzić jak się zachowa
4. (ad.5) Można podzielić serwer na kilka wirtualnych środowisk. Zdobycie roota w jednym z nich nie pozwoli na przejęcie kontroli nad pozostałymi.

OpenVZ

- Tworzy wirtualne środowiska (VE)
- Obsługuje Linuksa
- VE jest tylko 1-3% wolniejsze
- Checkpoint - możliwość przeniesienia VE na inny serwer bez potrzeby restartu
- obsługuje do 64 procesorów i 64 GB pamięci RAM
- Pojedyncze VE można przeskalować tak, by wykorzystywało wszystkie zasoby komputera

1. (ad.4) Checkpoint to zapisany stan VE

Virtuozzo

- Stworzony przez SWsoft
- Oparty na OpenVZ
- Jedno wspólne jądro
- Obsługuje Linuksa od 2001, Windows od 2005
- Wikipedia twierdzi, że jest to jedyny program tego typu obsługujący Windows
- Pozwala uruchomić setki VE zachowując funkcjonalność serwera
- Można przekierować zasoby do różnych VE bez konieczności restartu
- Zawiera narzędzia do zarządzania wieloma VE naraz

1. (ad.1) Jest to produkt komercyjny
2. (ad.4) Wersja dla Linuksa dostępna jest z licencją GNU General Public License

- Pełne emulowanie innej maszyny
- Zapewnia największą przenośność
- Duży spadek wydajności

1. (ad.1) Pełne emulowanie, czyli symulowanie działania procesora, pamięci i różnych urządzeń wejścia/wyjścia
2. (ad.2) Ponieważ nie korzystamy (zazwyczaj) ze wsparcia sprzętowego maszyny hosta, to emulator łatwo da się przenieść na inną maszynę
3. (ad.3) Konieczność emulowania wszystkich instrukcji docelowego procesora jak i innych elementów komputera znacząco zmniejsza szybkość działania emulatora

Bochs

- Emulator PC klasy x86 i amd64
- Kevin Lawton
- Pierwotnie program komercyjny
- Popularny wśród twórców systemów operacyjnych
- Emuluje wszystkie elementy komputera
- Istnieją wersje pod Windows, Linuksa oraz Mac OS X

1. (ad.3) W 2000 roku Mandrakesoft (Mandriva) wykupił program i wypuścił wersję dla Linuksa z licencją LGPL
2. (ad.4) Bochs jest preferowany przez twórców systemów operacyjnych, gdyż posiada raportowanie i zrzuty plików, których brakuje w innych emulatorach
3. (ad.5) Jak większość emulatorów nie posiada możliwości wirtualizacji procesora

DOSBox

- Emulator PC klasy x86
- Open source
- Tworzy środowisko DOS-owe
- Przeznaczony głównie do uruchamiania starych gier
- Wersje pod Linuksa, OpenBSD, FreeBSD, Windows, Mac OS X, OS/2, Palm OS, RISC OS oraz BeOS

1. (ad.1) Emulowana maszyna działa z szybkością około 10%
2. (ad.4) Na stronie DOSBoxa zamieszczona jest lista obsługiwanych gier
3. (ad.4) Niektórzy producenci wypuszczają paczki z DosBoxem i starą dosową grą - np. idSoftware zrobił to z Wolfensteinem 3D

QEMU
bez kqemu w systemie Hosta

- Fabrice Bellard
- Bez kqemu może pracować w trybie pełnej emulacji
- Pozwala emulować komputer klasy x86, amd64, alpha, mips oraz sparc
- Emuluje procesor jak i inne podzespoły docelowej maszyny
- Może służyć do uruchomienia wielu wirtualnych komputerów różnego typu na jednym komputerze
- Dynamiczna translacja przyspiesza szybkość emulacji
- Wersje pod Linuksa, Windows, Mac OS X oraz OpenSolaris

1. (ad.3) Dzięki pełnej emulacji nie ma potrzeby modyfikacji systemu operacyjnego gościa

Wine

- Wine Is Not an Emulator
- Bob Amstadt i Eric Youngdale
- Zamiast emulować, Wine zapewnia interfejs, którym można zastąpić jądro Windows
- Z małymi problemami uruchamia większość oprogramowania
- Posiada odpowiedniki większości bibliotek dll
- Moduł Direct3D jest ciągle rozwijany (np. dodawana jest obsługa Pixel Shaderów)
- Może używać oryginalnych bibliotek dll Microsoftu
- Napisany na Linuksa, ale są też wersja dla Mac OS X, FreeBSD oraz Solarisa

1. (ad.1) Rekurencyjny akronim uwidacznia często pomijany szczegół - Wine nie jest emulatorem. Pozwala jedynie uruchamiać aplikacje przeznaczone na Windows pod Linuksem
2. (ad.2) Projekt Wine powstał w 1993 jako mechanizm uruchamiania windowsowych aplikacji pod Linuksem. Pierwotnie kierowany był do aplikacji 16-bitowych (Win 3.x), ale obecnie obsługuje przede wszystkim 32-bitowe
3. (ad.2) Wersja beta została wydana dopiero w 2005 roku
4. (ad.2) Wine został wydany z licencją LGPL
5. (ad.4) Niektóre programy oczywiście uruchamiają się bez problemów, inne zaś wcale
6. (ad.7) Korzystanie z natywnych bibliotek dll dołączonych do Windows wymaga oczywiście posiadania na niego licencji

- Jeff Dike, Paolo Giarrusso (aka BlaisorBlade), Bill Stearns
- user-mode-linux.sourceforge.net (w przebudowie)
- umlwin32.sourceforge.net
- Pierwotnie na x86, porty na IA-64, AMD64, PowerPC

1. J. Dike jest twórcą, P. G. oraz B. S. mają bardzo duży wpływ, są w zasadzie współautorami.
2. User Mode Linux opiera się na pomyśle uruchomienia zmodyfikowanego jądra Linuksa jako zwykłego procesu użytkownika. Takie jądro „in userspace" korzysta z wydzielonego obrazu partycji (najczęściej zapisanego w pliku na partycjach Hosta).
3. Powstała implementacja(próba) UML na Windows, tzn. jądro linuksowe uruchomione jako proces Windows, ale korzystała z Cygwina do implementacji funkcjonalności; chłopaki chcieli to przepisać tak, by korzystało z WinAPI, ale szybko zrezygnowali i od 2002 projekt nie jest rozwijany.

- Serwery wirtualne (linode.com, od $19.95 / miesiąc; linuxzoo.net)
- Wirtualne sieci (z wyjściem na świat) (www.netkit.org) (VNUML)
- X – poprzez sieć mogą wyświetlać się na hoście
- honey-pot
- debugowanie jądra

1. Na linode.com można dostać konto z uprawnieniami roota, dostępnych jest kilka najpopularniejszych dystrybucji do wyboru.
2. linuxzoo.net oferuje darmowy dostęp do konta shellowego dla tych, którzy chcą się uczyć administrowania Linuksem, ale oni mają kilka komputerów z kilkunastoma maszynami wirtualnymi; jest quota na czas pracy; ale jak zajrzałem na tę stronę, to tylko 2 maszyny były zajęte.
3. Za pomocą UML i netkit.org można zbudować wirtualną sieć i testować programy sieciowe, taka sieć może mieć wyjście na świat, ale konfiguracja tego jest nietrywialna, zwłaszcza po stronie Hosta (instalacja dodatkowych sterowników i narzędzi).
4. Istnieje też inne rozwiązanie Virtual Network User Mode Linux oferujące podobne możliwości.
5. UML jest w sam raz do testowania 2. i 3. zadania z SO-Lab.

Zalety:
- qemu – szybkość
- Xen, VMWare ESX – prostota (instalacja, koncepcja)
- *hostfs* – banalnie łatwy dostęp do plików hosta
- „krówki" (pliki Copy On Write)
- **MADV_REMOVE** – należy wkrótce oczekiwać wparcia hotplug pamięci w UML

Wady:
- szybkość

1. UML jest szybszy niż qemu (bez kqemu) i innych emulatorów.
2. Instalacja UML jest banalnie prosta: wystarczy pobrać plik z jądrem skompilowanym do architektury UML, obraz systemu plików i już można używać UML, jedynie „usieciowienie" wymaga więcej pracy.
3. Host nie wymaga modyfikacji, chyba, że chcemy nałożyć łatki SKAS, o czym dalej.
4. UML działa w bardzo prosty sposób – wywołania systemowe przekazuje do Hosta, co jest mniej skomplikowane niż domeny Xena.
5. Ponieważ UML działa jako proces(y) na Hoście, można je kontrolować za pomocą standardowych narzędzi (top, ps, itp.)
6. UML# mount none /host -t hostfs -o /home/user
   montuje hostowy /home/user do /host na UML
7. UML oferuje pliki Copy On Write (cow), które pozwalają współdzielić jeden obraz systemu plików pomiędzy wiele instancji UML; z moich eksperymentów wynika, że się nie opłaca, bo pliki cow są większe niż sam obraz...

PTRACE_SYSEMU vs. PTRACE_SYSCALL

test	bez SYSEMU	z SYSEMU	zysk
getpid 10.000.000	real 6m16.461s user 0m5.077s sys 0m58.462s	real 3m55.052s user 0m5.962s sys 0m46.385s	60%
MySQL „run-all-tests"	real 18m38.329s user 2m37.370s sys 6m3.040s	real 13m57.353s user 2m17.370s sys 3m54.310s	33%
make bzImage modules	real 13m35.457s user 4m18.885s sys 1m40.481s	real 13m5.980s user 4m18.096s sys 1m40.115s	4%

Na podstawie: sysemu.sourceforge.net

1. UML wykorzystuje ptrace() z flagą SYSCALL do monitorowania wywołań systemowych, ten sposób jednak ma wady – UML podmienia syscalla na wywołanie getpid() a następnie wykonuje właściwą funkcję i podstawia jej wynik, co jest nieopłacalne; dlatego też napisano łatkę SYSEMU.
2. ptrace() z PTRACE_SYSEMU nie wywołuje zbędnego syscalla (tego z getpidem()).
3. Zyski przedstawione na stronie są naciągane (liczone na podstawie realtime), ale i tak jest nieźle.
4. getpid() natywnie: niecałe r:8, u:4, s:4[s].

SKAS
Separate Kernel Address Space

- W waniliowym Linuksie 2.6.0+ SKAS1
- SKAS3 v8.2 dla Linuksa 2.6.20-
- Szybciej i bezpieczniej niż z Traced Thread
- *[...]kernel build is almost twice as fast with skas mode as with tt mode, and is within 30% of the host's time. Bill Stearns saw a script's running time drop from 50 seconds to 14 seconds, almost a quadrupling in speed.*
- *It's SMP-safe, and it's running stably on SMP hosts.*

1. W TT procesy UML i jądro UML współdzielą tą samą przestrzeń adresową – mogą uzyskać niebezpiecznie duże uprawnienia, w skrajnych przypadkach – wydostać się poza UML do hosta
2. TT:
   - każdy proces UML jest odwzorowywany na proces Hosta
   - Tracing Thread śledzi ich syscalle i przekierowuje do jądra UML; wykorzystuje w tym celu powolne sygnały
3. SKAS:
   - UML kernel thread na hoście
   - UML userspace (procesy UML)
   - 2 procesy odpowiadające za operacje wejścia-wyjścia
   - W sumie 4 procesy hosta per UML

- Procesor Very Long Instruction Word
- Intel Itanium (core: Merced)
- Porażka biznesowa
- Morfing kodu (programowy ale zrealizowany w krzemie)
- Można zmienić mikrokod już po wyprodukowaniu układu
- Tłumaczenie instrukcji odbywa się tylko raz; pętle i odwołania do tego samego fragmentu kodu są pobierane w postaci rozkazów VLIW z pamięci cache
- Optymalizacja tłumaczonego kodu (usuwanie zbędnych instrukcji!)
- Torvalds i inne sławy miały swój wkład w projekt...

1. Itanium również jest VLIW oraz korzysta z morfingu kodu
2. Mikrokod jest programem który tłumaczy Assembler x86 na wewnętrzny język procesora. Mikrokod można załadować do procesora, ale jest to rozwiązanie sprzętowe – OS nic nie wie o fakcie morfingu.
3. Przy pomocy morfingu kodu można sprawić, że jeden procesor z zewnątrz wygląda jakby implementował inną architekturę, co rodzi możliwość wykorzystania do emulacji; w praktyce jest to ekonomicznie nieopłacalne.

- Rootkit zaprojektowany przez Joannę Rutkowską w 2006
- Teoretycznie umożliwia „całkowite" i „niewykrywalne" przejęcie kontroli nad systemem ofiary
- Bazuje na AMD-V
- Kiedy system „połknie niebieską tabletkę":
  - Blue Pill przechwytuje system „w locie" i ładuje go do swojej VM
  - Użytkownik ofiary pracuje bez straty wydajności (...?) i bez możliwości wykrycia Blue Pill (...?)
- Utrata części zasobów systemowych na rzecz Blue Pill:
  - Czas procesora: można przestawiać zegar...
  - Pamięć: trzeba ukrywać rootkit i *jednocześnie* nie zmniejszyć rozmiaru dostępnej pamięci
- Prototypowa implementacja z opublikowanymi źródłami

1. Przechwycenie systemu w locie jest ponoć ogólnie wykonalne...generalnie eksperci się zgadzają, że to jest prostsza część pomysłu
2. System ofiary ładowany jest do maszyny wirtualnej rootkita, który staje się hypervisor'em