Linux VServer - wirtualizacja przez separację

Jan Rękorajski
baggins@mimuw.edu.pl

15 grudnia 2005

Dostępne techniki wirtualizacji

Emulatory

- QEMU, Bochs
- pełna emulacja procesora i sprzętu
- bardzo duży narzut, każda instrukcja musi zostać przetworzona

Maszyny wirtualne

- UML, Xen, VMware
- wirtualizacja sprzętu
- średni narzut, nadzorca przekazuje odwołania do urządzeń

Wirtualne przestrzenie

- VServer, BSD Jail, Solaris Zones
- separacja procesów
- praktycznie brak narzutu, jedno jądro, jeden system

Dostępne techniki wirtualizacji

Emulatory

- QEMU, Bochs
- pełna emulacja procesora i sprzętu
- bardzo duży narzut, każda instrukcja musi zostać przetworzona

Maszyny wirtualne

- UML, Xen, VMware
- wirtualizacja sprzętu
- średni narzut, nadzorca przekazuje odwołania do urządzeń

Wirtualne przestrzenie

- VServer, BSD Jail, Solaris Zones
- separacja procesów
- praktycznie brak narzutu, jedno jądro, jeden system

Dostępne techniki wirtualizacji

Emulatory

- QEMU, Bochs
- pełna emulacja procesora i sprzętu
- bardzo duży narzut, każda instrukcja musi zostać przetworzona

Maszyny wirtualne

- UML, Xen, VMware
- wirtualizacja sprzętu
- średni narzut, nadzorca przekazuje odwołania do urządzeń

Wirtualne przestrzenie

- VServer, BSD Jail, Solaris Zones
- separacja procesów
- praktycznie brak narzutu, jedno jądro, jeden system

Wstęp

Założenia

Separacja przestrzeni wykonania

- Procesy
- System plików
- Sieć

Inne ograniczenia

- Rozszerzenia montowań typu “bind”
- Standardowa, współdzielona quota
- Scheduler
- CPUSET
- Izolacja zasobów

Założenia

Separacja przestrzeni wykonania

- Procesy
- System plików
- Sieć

Inne ograniczenia

- Rozszerzenia montowań typu “bind”
- Standardowa, współdzielona quota
- Scheduler
- CPUSET
- Izolacja zasobów

Separacja przestrzeni procesów

Linux capabilities

- Rozdrobnienie uprawnień z tradycyjnego modelu wszystko (UID=0) albo nic (UID<>0) pozwalające na umożliwienie wykonywania poszczególnych czynności (np. CAP_CHOWN, CAP_NET_RAW, CAP_SYS_TIME).
- Większość zdjęta dla procesów wewnątrz vservera

Security contexts

- Podział procesów na grupy nie mające ze sobą kontaktu ani informacji o swoim istnieniu
- Domyślny kontekst (kontekst hosta) ma pełne uprawnienia
- Kontekst widzi (i może przesyłać wiadomości)

Separacja przestrzeni procesów

Linux capabilities

- Rozdrobnienie uprawnień z tradycyjnego modelu wszystko (UID=0) albo nic (UID<>0) pozwalające na umożliwienie wykonywania poszczególnych czynności (np. CAP_CHOWN, CAP_NET_RAW, CAP_SYS_TIME).
- Większość zdjęta dla procesów wewnątrz vservera

Security contexts

- Podział procesów na grupy nie mające ze sobą kontaktu ani informacji o swoim istnieniu
- Domyślny kontekst, (kontekst hosta, 0), może tworzyć inne konteksty
- Kontekst widzi (i może wpływać przez sieć) wszystkie konteksty

Separacja przestrzeni procesów

Linux capabilities

- Rozdrobnienie uprawnień z tradycyjnego modelu wszystko (UID=0) albo nic (UID<>0) pozwalające na umożliwienie wykonywania poszczególnych czynności (np. CAP_CHOWN, CAP_NET_RAW, CAP_SYS_TIME).
- Większość zdjęta dla procesów wewnątrz vservera

Security contexts

- Podział procesów na grupy nie mające ze sobą kontaktu ani informacji o swoim istnieniu
- Domyślny kontekst, (kontekst hosta, 0), może tworzyć inne konteksty
- Kontekst widza (1), widzi procesy we wszystkich kontekstach

Separacja przestrzeni procesów

Linux capabilities

- Rozdrobnienie uprawnień z tradycyjnego modelu wszystko (UID=0) albo nic (UID<>0) pozwalające na umożliwienie wykonywania poszczególnych czynności (np. CAP_CHOWN, CAP_NET_RAW, CAP_SYS_TIME).
- Większość zdjęta dla procesów wewnątrz vservera

Security contexts

- Podział procesów na grupy nie mające ze sobą kontaktu ani informacji o swoim istnieniu
- Domyślny kontekst, (kontekst hosta, 0), może tworzyć inne konteksty
- Kontekst widza (1), widzi procesy we wszystkich kontekstach

Separacja przestrzeni procesów

Linux capabilities

- Rozdrobnienie uprawnień z tradycyjnego modelu wszystko (UID=0) albo nic (UID<>0) pozwalające na umożliwienie wykonywania poszczególnych czynności (np. CAP_CHOWN, CAP_NET_RAW, CAP_SYS_TIME).
- Większość zdjęta dla procesów wewnątrz vservera

Security contexts

- Podział procesów na grupy nie mające ze sobą kontaktu ani informacji o swoim istnieniu
- Domyślny kontekst, (kontekst hosta, 0), może tworzyć inne konteksty
- Kontekst widza (1), widzi procesy we wszystkich kontekstach

chroot(1)

Standardowy

- Zmienia katalog główny dla procesu i jego potomków
- Informacja wewnątrz jądra jest ulotna, kolejne wywołania chroot zamazują informację o aktualnym /.
- Łatwo z niego wyjść np. chroot(../..)

Atrybut –barrier

Uniemożliwia wykonanie chroot do katalogu na którym jest ustawiony.

chroot(1)

Standardowy

- Zmienia katalog główny dla procesu i jego potomków
- Informacja wewnątrz jądra jest ulotna, kolejne wywołania chroot zamazują informację o aktualnym /.
- Łatwo z niego wyjść np. chroot(../..)

Atrybut –barrier

Uniemożliwia wykonanie chroot do katalogu na którym jest ustawiony.

Przestrzenie nazw VFS

Dostępna implemetacja

"Widok" na zamontowane systemy plików, standardowo jeden dzielony przez wszystkie procesy

Modyfikacje

VServer wprowadza możliwość tworzenia osobnych przestrzeni nazw dla poszczególnych kontekstów, dzięki czemu wszelkie modyfikacje VFS, takie jak mount, nie są widoczne w innych kontekstach.

Przestrzenie nazw VFS

Dostępna implemetacja

"Widok" na zamontowane systemy plików, standardowo jeden dzielony przez wszystkie procesy

Modyfikacje

VServer wprowadza możliwość tworzenia osobnych przestrzeni nazw dla poszczególnych kontekstów, dzięki czemu wszelkie modyfikacje VFS, takie jak mount, nie są widoczne w innych kontekstach.

Znakowanie plików

Zastosowanie

- Nie jest wymagane do działania vservera
- Pozwala na pełną izolację kontekstów na poziomie VFS
- Wymagane do działania limitów dyskowych dla kontekstów
- Wymagane do działania kontekstowej quoty na współdzielonej partycji

Znakowanie plików

Implementacja

- Numer kontekstu zapisywany jest w każdym i-węźle jako XID
  - w wyższych bitach dostępnych pól (UID, GID), co niestety redukuje rozmiar tychże do 16/24 bitów
  - w nieużywanym miejscu wewnątrz i-węzła (ext2/3, reiserfs, xfs)
- Wszystkie testy dostępu do i-węzłów zostały rozszerzone o sprawdzanie ID kontekstu
- Wyjątki: kontekst hosta i widza
- Nieoznakowanie pliki są traktowane jakby należały do aktualnego kontekstu
- Plik modifykowany wewnątrz kontekstu automatycznie zostaje oznakowany ID tego kontekstu

Unifikacja

- Współdzielenie plików między kontekstami przez twarde linki
- Oszczędność miejsca na dysku, buforów dyskowych, pamięci dzielonej
- Złośliwy kontekst może jednak takie pliki “popsuć”
- Rozwiązanie dostępne - atrybut –immutable
- Chcemy jednak móc takie pliki kasować, stąd zmodyfikowany atrybut –iunlink
- Nowe rozwiązanie - COW links (kopiowanie przy zapisie), podczas modyfikacji link jest zrywany i plik jest kopiowany niezauważalnie dla użytkownika

Zabezpieczenie ProcFS

- Nie chcemy żeby wszystkie elementy systemu plików proc były widoczne z każdego kontekstu
- Każdemu elementomi systemu plików proc można przypisać flagi: Admin, Watch i Hide.
- Hide ukrywa całkowicie element
- Admin zezwala na dostęp dla kontekstu hosta
- Watch zezwala na dostęp dla kontekstu vservera

Sieć IPv4

chbind

Wywołanie systemowe set_ipv4root ogranicza widoczność urządzeń sieciowych i konkretnych adresów IP dla vservera

NGNET

Nowa implementacja wirtualizacji sieci dla vservera

- Kontekst NetworkID, pełna separacja urządzeń i adresów sieciowych
- Urządzenie imaq czyli loopback transmitujący pakiety między kontekstami

Sieć IPv4

chbind

Wywołanie systemowe set_ipv4root ogranicza widoczność urządzeń sieciowych i konkretnych adresów IP dla vservera

NGNET

Nowa implementacja wirtualizacji sieci dla vservera

- Kontekst NetworkID, pełna separacja urządzeń i adresów sieciowych
- Urządzenie vnet czyli loopback transmitujący pakiety między kontekstami

Sieć IPv4

chbind

Wywołanie systemowe set_ipv4root ogranicza widoczność urządzeń sieciowych i konkretnych adresów IP dla vservera

NGNET

Nowa implementacja wirtualizacji sieci dla vservera

- Kontekst NetworkID, pełna separacja urządzeń i adresów sieciowych
- Urządzenie vnet czyli loopback transmitujący pakiety między kontekstami

Sieć IPv4

chbind

Wywołanie systemowe set_ipv4root ogranicza widoczność urządzeń sieciowych i konkretnych adresów IP dla vservera

NGNET

Nowa implementacja wirtualizacji sieci dla vservera

- Kontekst NetworkID, pełna separacja urządzeń i adresów sieciowych
- Urządzenie vnet czyli loopback transmitujący pakiety między kontekstami

Dodatki - System plików

Rozszerzenia montowań typu “bind”

- mount –bind pozwala zamontować część systemu plików w innym miejscu
- Zwykły mount –bind dziedziczy wszelkie opcje po montowanym fs
- Vserver daje możliwość zamontowania z opcjami ro, noatime, nodiratime

Dodatki - System plików

Rozszerzenia montowań typu “bind”

- mount –bind pozwala zamontować część systemu plików w innym miejscu
- Zwykły mount –bind dziedziczy wszelkie opcje po montowanym fs
- Vserver daje możliwość zamontowania z opcjami ro, noatime, nodiratime

Dodatki - System plików c.d.

Standardowa, współdzielona quota

- Ze względów bezpieczeństwa nie chcemy mieć wewnątrz vservera plików urządzeń
- Quota do działania potrzebuje jednak dostępu do rzeczywistego urządzenia na którym jest system plików
- Rozwiązaniem jest urządzenie-przekaźnik vroot które potrafi obsłużyć jedynie wywołania quotactl i przekazać je do właściwego urządzenia

Dodatki - System plików c.d.

Standardowa, współdzielona quota

- Ze względów bezpieczeństwa nie chcemy mieć wewnątrz vservera plików urządzeń
- Quota do działania potrzebuje jednak dostępu do rzeczywistego urządzenia na którym jest system plików
- Rozwiązaniem jest urządzenie-przekaźnik vroot które potrafi obsłużyć jedynie wywołania quotactl i przekazać je do właściwego urządzenia

Dodatki - Scheduler

Token Bucket

- Ogranicza użycie czasu procesora przez kontekst
- Wiadro o rozmiarze S jest wypełniane określoną ilością żetonamów R co czas T, dopóki nie zostanie wypełnione. Przy każdym cyklu zegara działający proces pochłania dokładnie jeden żeton, gdy wiadro zostanie opróżnione, proces jest wstrzymywany dopóki nie pojawi się co najmniej M żetonów.

Cpuset

- Pozwala na przypisanie zbioru procesorów i węzłów pamięci dla zbioru procesów
- Skrypty do obsługi vserverów posiadają wsparcie do konfigurowania i korzystania z Cpusets

Dodatki - Scheduler

Token Bucket

- Ogranicza użycie czasu procesora przez kontekst
- Wiadro o rozmiarze S jest wypełniane określoną ilością żetonamów R co czas T, dopóki nie zostanie wypełnione. Przy każdym cyklu zegara działający proces pochłania dokładnie jeden żeton, gdy wiadro zostanie opróżnione, proces jest wstrzymywany dopóki nie pojawi się co najmniej M żetonów.

Cpuset

- Pozwala na przypisanie zbioru procesorów i węzłów pamięci dla zbioru procesów
- Skrypty do obsługi vserverów posiadają wsparcie do konfigurowania i korzystania z Cpusets

Dodatki - Zasoby

Ponieważ większość zasobów jest współdzielona przez różne konteksty konieczna jest ich dodatkowa izolacja ze względów bezpieczeństwa i poprawnego rozliczania (accounting).

- pamięć dzielona, IPC
- numery użytkowników, grup i procesów
- pseudoterminale (Unix ptys)
- gniazda sieciowe

Wstęp

Zalety i wady

Zalety

- Prostota instalacji i obsługi
- Współdzielenie zasobów
- Wydajność
- Bezpieczeństwo

Wady

- Bezpieczeństwo?
- Jedno jądro systemu

Zalety i wady

Zalety

- Prostota instalacji i obsługi
- Współdzielenie zasobów
- Wydajność
- Bezpieczeństwo

Wady

- Bezpieczeństwo?
- Jedno jądro systemu

Zastosowania

- Prywatne serwery wirualne sprzedawane przez dostawców usług
- Separacja (konfliktujących) serwisów
- Zwiększenie bezpieczeństwa, host bez dostępu do sieci uruchamia vserver z wszystkimi usługami, pozwala to na pełny audyt środowiska
- Uproszczenie obsługi, łatwiej przenieść vserver na inną maszynę niż cały system
- Odporność na awarie (fail-over)
- Testowanie, vserver daje bardziej realistyczne środowisko niż chroot przy instalacji wielu różnych dystrybucji na jednej maszynie

Zastosowania

- Prywatne serwery wirualne sprzedawane przez dostawców usług
- Separacja (konfliktujących) serwisów
- Zwiększenie bezpieczeństwa, host bez dostępu do sieci uruchamia vserver z wszystkimi usługami, pozwala to na pełny audyt środowiska
- Uproszczenie obsługi, łatwiej przenieść vserver na inną maszynę niż cały system
- Odporność na awarie (fail-over)
- Testowanie, vserver daje bardziej realistyczne środowisko niż chroot przy instalacji wielu różnych dystrybucji na jednej maszynie

Zastosowania

- Prywatne serwery wirualne sprzedawane przez dostawców usług
- Separacja (konfliktujących) serwisów
- Zwiększenie bezpieczeństwa, host bez dostępu do sieci uruchamia vserver z wszystkimi usługami, pozwala to na pełny audyt środowiska
- Uproszczenie obsługi, łatwiej przenieść vserver na inną maszynę niż cały system
- Odporność na awarie (fail-over)
- Testowanie, vserver daje bardziej realistyczne środowisko niż chroot przy instalacji wielu różnych dystrybucji na jednej maszynie

Zastosowania

- Prywatne serwery wirualne sprzedawane przez dostawców usług
- Separacja (konfliktujących) serwisów
- Zwiększenie bezpieczeństwa, host bez dostępu do sieci uruchamia vserver z wszystkimi usługami, pozwala to na pełny audyt środowiska
- Uproszczenie obsługi, łatwiej przenieść vserver na inną maszynę niż cały system
- Odporność na awarie (fail-over)
- Testowanie, vserver daje bardziej realistyczne środowisko niż chroot przy instalacji wielu różnych dystrybucji na jednej maszynie

Zastosowania

- Prywatne serwery wirualne sprzedawane przez dostawców usług
- Separacja (konfliktujących) serwisów
- Zwiększenie bezpieczeństwa, host bez dostępu do sieci uruchamia vserver z wszystkimi usługami, pozwala to na pełny audyt środowiska
- Uproszczenie obsługi, łatwiej przenieść vserver na inną maszynę niż cały system
- Odporność na awarie (fail-over)
- Testowanie, vserver daje bardziej realistyczne środowisko niż chroot przy instalacji wielu różnych dystrybucji na jednej maszynie

Zastosowania

- Prywatne serwery wirualne sprzedawane przez dostawców usług
- Separacja (konfliktujących) serwisów
- Zwiększenie bezpieczeństwa, host bez dostępu do sieci uruchamia vserver z wszystkimi usługami, pozwala to na pełny audyt środowiska
- Uproszczenie obsługi, łatwiej przenieść vserver na inną maszynę niż cały system
- Odporność na awarie (fail-over)
- Testowanie, vserver daje bardziej realistyczne środowisko niż chroot przy instalacji wielu różnych dystrybucji na jednej maszynie

Pokaz

- Instalacja
- Uruchamianie/zatrzymywanie/restart
- Kontekst widza
- Przestrzenie nazw VFS
- Quota na współdzielonej partycji (vroot)
- Limity dyskowe
- CPUSET
- Token bucket

Wstęp

- Główna strona projektu
  `http://linux-vserver.org/`
- Narzędzia do obsługi Vserverów
  `http://savannah.nongnu.org/projects/util-vserver/`
- Nowa implementacja API Linux-Vservera
  `http://dev.gentoo.org/~hollow/vserver/libvserver/`
- Eksperymentalne łatki
  `http://vserver.13thfloor.at/Experimental/`