

Przegląd technik wirtualizacji i separacji w nowoczesnych systemach rodziny UNIX

CONFidence 2005

IX Liceum Ogólnokształcące im. C.K. Norwida w Częstochowie
Krajowy Fundusz na Rzecz Dzieci
Wojciech A. Koszek
dunstan@FreeBSD.czyst.pl

Plan prezentacji:

- (bardzo) krótkie omówienie zagrożeń związanych z oprogramowaniem
- Przedstawienie potencjalnych dróg włamania
 - obraz sytuacji włamania z punktu widzenia systemu operacyjnego i aplikacji
 - wprowadzenie do metod separacji niebezpiecznego oprogramowania
 - omówienie technik wirtualizacji
- Podsumowanie

Zagrożenia związane z oprogramowaniem:

- ataki wykorzystujące szczególne aspekty uruchamiania kodu na danej architekturze
- zaufanie uruchamianej aplikacji do danych niezależnych od początkowych założeń (protokoły, interfejsy, wszelkie aspekty komunikacji systemowej)
- popularyzacja WWW, wiedzy dotyczącej ataków z wykorzystaniem technik typu buffer overflow, format-string, XSS, injection..

Sytuacje najczęstsze:

- Dostęp do bazy danych/plików/katalogów poprzez skrypty PHP z rozszerzeniami:
 - skrypt odbiera wartość od użytkownika i bez uprzedniej walidacji wykorzystuje je kontynuując typowe działanie
 - możliwość przekazania dodatkowych ciągów "../", "./" może prowadzić do ciekawych rezultatów
 - skrypty stworzone przez pomysłowych programistów mogą uruchamiać dodatkowe programy (shell/Perl) korzystając z przekazanych argumentów
- Próby wykorzystywania znanych luk w oprogramowaniu (zarówno na aplikacje internetowe jak i systemowe)

Sytuacje rzadsze: próba ataku na konkretną implementację:

- aplikacji systemowej: poprzez audyt, analizę działania, eksperymenty
 - biblioteki: podatne stają się wszystkie programy zlinkowane z biblioteką
 - systemu: interfejsy komunikacji z aplikacją, jądrem systemu, usługami zdalnymi
 - protokołu
- Ograniczeniem jest jedynie wiedza atakującego.

Cechy wspomnianych ataków:

- wszystkie usługi koegzystują korzystając z zasobów głównej maszyny:
 - system plików
 - pamięć
 - dostęp do procesora
- wszystkie współdzielą obiekty systemu operacyjnego (deskryptory plików, procesy itd).
- atak na pojedynczą usługę powoduje (pośrednio) zagrożenie innych usług: local/remote DoS

Możliwe rozwiązania:

- znalezienie bezpiecznych zamienników popularnych usług
- audyt niebezpiecznych aplikacji
- separacja oprogramowania

Odseparowanie aplikacji od głównego systemu:

- atak na odseparowany demon daje dostęp do limitowanego środowiska
- w razie ataku:
 - usunięte zostają jedynie kopie plików potrzebnych aplikacji do działania
 - włamywacz z "osobnym" UID $\neq 0$ ma trudne zadanie
- wymagane wsparcie od strony systemu operacyjnego

Odseparowane środowisko dla aplikacji:

- pliki binarne
- pliki danych
- biblioteki dzielone
- struktura katalogów
- pliki specjalne (*/proc/**, */dev/**)

Problemy dotyczące separacji:

- wymagane wsparcie od strony systemu operacyjnego
- konieczność duplikacji zasobów
 - rozwiązania poprzez NULLFS i alternatywy
- utrzymywanie aktualności oprogramowania jest utrudnione

Separacja kiedyś:

- *nix: *chroot()*:
 - ✓ limitowanie widoczności systemu plików
 - ✗ Brak separacji w warstwie procesów
 - ✗ brak dodatkowych limitów obejmujących zasoby systemowe

Możliwe strategie:

- selektywne udostępnianie kluczowej funkcjonalności
 - dodatkowe warstwy bezpieczeństwa (TrustedBSD, SELinux, Trusted {IRIX, Solaris})
- duplikacja krytycznych obiektów
- autonomiczne środowisko w głównym systemie: zminimalizowana interakcja z systemem głównym
- wirtualna maszyna: pełna emulacja maszyny

FreeBSD - jail():

- Limitowanie systemu plików (wewnętrznie korzysta z chroot())
- limitowanie w warstwie procesów:
 - ograniczone IPC
- system przystosowany do zarządzania instancjami oprogramowania

Solaris: Zones

- zintegrowane środowisko do konfiguracji (*zonecfg*)
 - dziedziczenie katalogów z głównego systemu plików
- możliwość konfiguracji poszczególnych, odseparowanych środowisk
- do niedawna problemy z bootowaniem na x86 ("*newboot*")

Wymienione mechanizmy nie działają gdy dochodzi do ataków:

- na kod odpowiedzialny za ich implementacje
- na główny system operacyjny
- wykorzystujących luki w odseparowanych zasobach

Rozbudowane możliwości dodatkowej kontroli:

- systemy posiadające wsparcie dla szczegółowej kontroli:
 - TrustedBSD
 - SEDarwin
 - SELinux
- Komercyjne systemy typu *-Trusted* (Solaris, IRIX)

Wirtualna maszyna: emulacja działającego komputera

- emulatory (Plex86 - prawdopodobnie pierwszy raz wykorzystano w nich PVI architektury x86)
- statyczne przepisywanie instrukcji
- wstawienie instrukcji niezdefiniowanych i wychwytywanie ich poprzez mechanizm Ptrace - uruchamianie BSD w user-space
- Alpine

Xen: innowacyjne podejście do technologii wirtualizacji:

- zmiana kontekstu wirtualizacji - już nie proces, ale cały system.
- co jest potrzebne systemowi do działania (obsługa pułapek, przerwań, wyjątków, przełączanie kontekstu procesów, zachowywanie rejestrów, wsparcie dla SMP, detekcja PCI/ISA, rozpoznawanie sprzętu)
- normalnie wywołania BIOSu, *int 0xXY* - w Xen'ie - wołania do HYPERVISOR'a, nadrzędnej architektury odpowiedzialnej za dostęp do fizycznych zasobów maszyny
- konieczność przepisania niskopoziomowej warstwy systemu.
- x Wady: obecnie jedynie x86.

Xen.. cd:

- Domena 0/X w dla Linux, NetBSD
- Domeny X dla FreeBSD (problemy w wersji *-unstable*, domena 0 koncentrowana wokół Linux)
- Eksperymentalne wsparcie dla Solaris

Referencje:

- FreeBSD
 - *<http://www.FreeBSD.org/>*
 - *<http://www.TrustedBSD.org>*
- SELinux
 - *<http://www.NSA.gov/selinux/>*
- Solaris
 - *<http://blogs.sun.com/{comay,dp,jbeck,menno,jclingan}>*

Dziękuję za uwagę

<http://security.proidea.org.pl>