Przegląd technik wirtualizacji i separacji w nowoczesnych systemach rodziny UNIX

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

CONFidence 2005
Kraków, 15-16.10.2005

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 separacji i wirtualizacji

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 wiedzy dotyczącej ataków z wykorzystaniem technik typu buffer overflow, format-string, race condition, XSS, SQL/HTML injection..

Sytuacje najczęstsze:

- Dostęp do bazy danych/plików/katalogów poprzez skrypty CGI:
  - 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: próby ataku na konkretną implementację:

- aplikacji systemowej: poprzez audyt, analizę działania, eksperymenty
- biblioteki: podatne stają się wszystkie programy linkowane 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, parametry systemu..)
- atak na pojedynczą usługę powoduje zagrożenie innych usług i maszyny

Możliwe rozwiązania:

- znalezienie bezpiecznych zamienników popularnych usług
- audyt niebezpiecznych aplikacji
- zminimalizowanie ryzyka w przypadku wykorzystania luk w oprogramowaniu
  - separacja
  - wirtualizacja

Separacja - osobne środowisko dla aplikacji:

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

Odseparowanie aplikacji od głównego systemu:

- atak na odseparowany demon daje dostęp do limitowanego środowiska
- w razie udanego ataku:
  - usunięte zostają jedynie kopie plików potrzebnych aplikacji do działania
  - włamywacz z "osobnym" UID != 0 ma utrudnione zadanie

Problemy dotyczące separacji:

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

Separacja kiedyś: *chroot()*

- limitowanie widoczności systemu plików
- brak separacji w warstwie procesów
- brak dodatkowych limitów obejmujących zasoby systemowe

Obecne zapotrzebowanie:

- selektywne udostępnianie kluczowej funkcjonalności
  - dodatkowe warstwy bezpieczeństwa zintegrowane z systemem operacyjnym
- duplikacja krytycznych obiektów
- autonomiczne środowisko: zminimalizowana interakcja z systemem głównym
- wirtualna maszyna: pełna emulacja maszyny

FreeBSD - jail():

- limitowanie systemu plików (mechanizm podobny do *chroot()*)
- limitowanie w warstwie procesów:
  - ograniczony zakres widoczności procesów i 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"*)

Rozbudowane mechanizmy bezpieczeństwa:

- systemy posiadające wsparcie dla obowiązkowej kontroli (MAC):
  - TrustedBSD
  - SEDarwin
  - SELinux (i wiele innych)
- Komercyjne systemy typu "*Trusted*" (Solaris, IRIX)

Wymienione mechanizmy nie działają gdy dochodzi do ataków:

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

Wirtualna maszyna: emulacja działającego komputera

- Emulatory
  - Bochs
  - 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

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 sprzętu, obsługa PCI/ISA)
- 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.
- 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/