(Ne)bezpečné programovanie

Peter Košinár

kosinar[zavináč]eset.sk

Eskalácia privilégií

- X je menej privilegovaný ako Y:
  - Klient < Server.
  - Bežný používateľ < root / administrator.
  - Užívateľský proces < kernel.
- X dokáže s Y komunikovať (vstup-výstup, sieť, ...)
- Ak X dokáže Y vhodným spôsobom zmiasť, môže jeho privilégia získať, alebo využiť.

Eskalácia privilégií - Linux

Linuxový rýchlokurz privilegovaného procesu:

- Proces beží pod nejakým užívateľom (UID) a skupinou (GID).
- UID=0 (zvyčajne zvaný „root“) je najvyšší šéf.
- UID, GID regulujú prístup k súborom, iným procesom, potenciálne sa podľa nich dá filtrovať sieťová komunikácia, ...
- Príkaz *id* o tom vie čo-to povedať.
- Ak má spustiteľný súbor nastavený SUID bit, pri spustení beží s privilégiami vlastníka, nie (len) spúštateľa.
- Ak má nastavený SGID bit, dostane (aj) skupinu vlastníka.

Logické chyby

- Najnebezpečnejší druh chýb – program niečo naozaj má umožňovať, ale umožňuje to aj niekomu, kto na to nie je oprávnený.
- Nedá sa im príliš vyhnúť technickými prostriedkami – „uhádnuť“ úmysel je doména veštcov, nie programátorov ☺
- Implementácia korešponduje s dizajnom, ale dizajn je zlý.
- Príklady:
    - PHP skript, ktorý „adminskosť“ určuje na základe toho, či je v požiadavke napísané „&admin=1”
    - Domáce routre a podobné zariadenia, ktoré bežne chcú heslo, ale umožňujú si celú konfiguráciu vypýtať bez autentifikácie.

Čítanie

- Program občas niečo vie, akurát to z neho treba vytĺcť.
- Bežné zdroje:
  - Dôverovanie informáciam, ktoré dodala nedôveryhodná strana.
  - Neinicializovaná alebo nevyčistená pamäť (šifrovacie kľúče, ...).
  - Občas priame čítanie z ľubovoľnej pamäte.
- Áno, o tomto bol aj onen slávny Heartbleed ☺

Zápis

- Najbežnejšie prepisované miesto – zásobník (návratové adresy, ...).
- Zmenená návratová adresa = program začne vykonávať, čo od neho chceme my.
- Pointre do kódu sú ale aj inde – malloc/free pamäťové bloky, C++ a tabuľky virtuálnych metód, ...
- Aj pointer na dáta môže byť užitočný – napríklad na prepísanie viac dát.

Obranné mechanizmy

- Limitovanie zraniteľností a ziskov z nich plynúcich:
  - Vypnuté sieťové služby
  - Linux capabilities, seccomp, ...
  - Windows sessions, jobs, ...
- Kompilátorom podporované kontroly v rámci programu:
  - Stack canaries, double-free detection, ...

Obranné mechanizmy

- NX / DEP = Nespúšťateľnosť dátových stránok:
  - Špeciálne relevantné pre stack a allokácie na heape.
  - Stále ale možno spúšťať existujúci kód.
  - Čoraz častejšie sú data-driven programy – Java, Javascript, …
  - Just-in-Time kompilácia je tiež dobrým zdrojom vlastného kódu.

- ASLR = Vnesenie prvku náhodnosti:
  - Samotná chyba sa síce môže prejaviť, ale útočník ju nemusí vedieť vp väčšine prípadov využiť.
  - Je to síce security-through-obscurity, ale 2^25 sekúnd > 1 rok.

Ďakujem za pozornosť

Peter Košinár

kosinar[zavináč]eset.sk