Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tadej Logar

Peskovniki

seminarska naloga

sistemska programska oprema

Ljubljana, 2021

Kazalo

Poglavje 1

Uvod

Pri delu z računalniško programsko opremo se velikokrat srečamo z problematiko programske opreme, ki je lahko zlonamerna. Za takšno opremo želimo imeti nadzorovano izvajalno okolje, za ta namen je pa koncept peskovnika zelo uporaben.

Z peskovnikom lahko sistematično omejimo aplikacijo, njeno izvajanje, ter podatke, s katerimi sme obdelovati.

Izboljševanje varnosti v računalniških sistemih je ena od glavnih, ampak ne edinih nalog peskovnikov.

Velikokatera implementacija peskovnika temelji na virtualizaciji, torej ustvarjanje navideznega okolja. Več o virtualizaciji bomo spoznali v prvem poglavju.

Poglavje 2

Virtualizacija

Računalniški sistem je sestavljen iz veliko delov strojne ter programske opreme, potrebnih za izvajanje sodobnih aplikacij. Virualizacija nam omogoča, da odklopimo posamezen del od celote, tako da ustvarimo navidezno različico tega. Ta navidezni del ima nadzorovan in omejen dostop do preostalih virov na računalniku. [8]

Virtualizacijo lahko dosežemo z uporabo veliko različnih tehnologij, vse od tistih na nivoju strojne opreme, kot so dodatni ukazi v procesorju in uporaba Hipervizorja za ustvarjanje navideznega stroja, do ustvarjanja navideznih sestavin operacijskih sistemov in izoliranja vsake aplikacije od druge. [8]

Poglavje 3

Peskovniki

Peskovnik deluje na principu izoliranja kode ali aplikacije od širšega računalniškega sistema. S tem peskovnik doseže omejitev dostopa do podatkov in ostalih virov. Tudi izvajanje kode je lahko onemogočeno, kot npr. izvajanje nekaterih sistemskih klicev. Vsi podatki, ki jih koda nujno potrebuje, morejo biti dostopni v peskovniku.

Sicer pa so peskovniki širok spekter različnih implementacij in namenov. Lahko popolno izolirajo vsak del računalniškega sistema posebej (qubesOS), lahko pa peskovnik skozi operacijski sistem ustvari izolirano okolje, kjer si program deli z ostalimi le jedro operacijskega sistema. Slednji ima več imen, kot so vsebniki (angl. container) in ječe (angl. jail). Še drugi pa so kot npr. peskovniki v spletnih brskalnikih, in pa kot peskovnik v operacijskem sistemu Android, ki izolira vsako posamezno aplikacijo.

Čeprav je praktično vsaka virtualizacija tudi peskovnik, obstajajo tudi peskovniki, ki niso zasnovani le na virtualizaciji. Primer takšnega peskovnika je seccomp (secure computing mode).

Peskovniki niso le uporabni samo za varnost, omogočajo lahko tudi obravnavo aplikacij, ki zaradi zastarele kode ali napak ustvarijo težave v operacijskem sistemu. Na primer, če je aplikacija zasnovana tako, da za delovanje potrebuje točno določen file descriptor, ali pa točno določen procesni identifikator (PID), potem je peskovnik možna rešitev.

Poglavje 4

Implementacija različnih peskovnikov

4.1 FreeBSD ječe

Ječa je implementacija virtualizacije na nivoju operacijskega sistema. Eno skupno jedro omogoča in zagotovi kompartmentalizacijo sistemskih virov. Ječa dobi svoje omrežne vire, datotečni sistem, ter svojo množico PID/UID/GID, in še več. Jedro omeji dostop posameznih ječ do širšega računalniškega sistema. [5]

Torej je vsaka ječa navidezno okolje, ki lahko ima svoje datoteke, procese in tudi uporabnike. Interakcije med ječami so omejene in vsaka ječa ima dostop do omrežnih virov točno določen in omejen.

Koncept ječe izvira iz sistemskega klica chroot, ampak je dandanes naprednejši in ima svoje sistemske klice, kot je jail(2). [6]

4.2 Linux container

Linux container oz. vsebnik ima veliko podobnega z FreeBSD ječo. Tudi vsebnik je virtualizacija na nivoju operacijskega sistema, kjer si vsebniki delijo eno jedro.

Ampak razlika je v jedru in kaj ponuja. Jedro namreč ne ponuja načina za ustvarjanje vsebnika, le vmesnik preko katerega se lahko ustvarijo.

FreeBSD ječa vsebuje le nekatere vire operacijskega sistema, ki si jih lasti, Linux vsebnik pa poskusi ustvariti povsem nov uporabniški svet (angl. userspace/userland), torej vso izvajalno kodo operacijskega sistema, ki teče izven jedra.

Linux vsebnik ne ustvari jedro, ampak za to namenska programska oprema, kot je Docker, podman, lxc, ...

4.3 seccomp

Kratica za secure computing mode, to je del Linux jedra, ki omogoča procesu vstop v varno stanje, kjer lahko izvaja le štiri sistemske klice na že odprtih file descriptorjih. To so exit(), sigreturn(), read() in write(). Pri drugačnem obnašanju, jedro ustavi proces. [7]

Seccomp ne uporablja virtualizacije za peskovnik, le popolno izolacijo posameznega procesa. Izvaja se preko sistemskega klica seccomp(2). [7]

4.4 Android Application Sandbox

Linux jedro omogoča več varnostnih funkcij, med njimi model za dodeljevanje dovoljenj na nivoju uporabnikov. Android Application Sandbox je peskovnik, ki za svojo delovanje uporabi prav ta model za izolacijo vseh aplikacij. [1]

Vsaki aplikaciji se dodeli unikaten uporabniški identifikator, UID, in aplikacija se zažene v svojem procesu. Jedro poskrbi za potrebno izolacijo. V Androidu se vse aplikacije nad jedrom, torej tudi aplikacije operacijskega sistema, nahajajo v takšnem peskovniku. [2]

4.5 Chromium Sandbox za Windows

Chromium je spletni brskalnik in vsebuje knjižnico, katera omogoča ustvarjanje procesov v peskovniku. Ti procesi imajo zelo okrnjeno izvajalno okolje, in njihova dovoljenja se lahko spreminjajo glede na izdana pravila. Dostop do datotečnega sistema je prepovedan, kot tudi ustvarjanje oken.

Chromium doseže peskovnik z uporabo funkcij, ki jih poda operacijski sistem. Opredeljuje dva tipa procesa, nadzorovalnika, imenovan broker, in target, proces v peskovniku.

Knjižnico za peskovnik mora povezati v oba tipa. [3][4]

4.5.1 Broker

V spletnem brskalniku je broker vedno proces brskalnika. Ustvarja in nadzoruje procese, ki se izvajajo v peskovniku, jim določa pravila in omogoča medprocesno komunikacijo.

4.5.2 Target

Tarča oz. target je proces, ki ga brskalnik izvaja v peskovniku. Hkrati je lahko več tarč.

Poglavje 5

Windows Sandbox

Windows Sandbox ustvari navidezno namizno okolje, kjer se lahko izvajajo poljubne aplikacije v izolaciji. To namizno okolje je začasno, in se ob zaustavitvi popolnoma izbriše, ter nima vpliva na izvajalno okolje, kjer se je ustvarilo.

Deluje s pomočjo virtualizacije strojne opreme. Z uporabo hipervizorja se ustvari novo Windows jedro, ki je izolirano od izvajalnega.

Aplikacije se ne delijo med gostiteljem in peskovnikom, kot pri navideznih strojih se mora namestiti zaželena aplikacija v peskovnik.

V resnici Windows Sandbox je navidezni stroj, ki potrebuje že nameščen Windows iz katerega se lahko vzpostavi. Ne potrebuje tiste sistemske datoteke iz gostiteljske namestitve, ki se ne spreminjajo skozi čas.[9][10]

5.1 Deljenje spomina

Gostiteljski sistem in vsi Windows Sandbox peskovniki si, zato ker so praktično enaki sistemi, neposredno delijo pomnilniški prostor, uporabljajo se enake pomnilniške strani.[9][10]

5.2 Načrtovalnik izvajanja

Izvajanje ali ne–izvajanje navideznih strojev poteka preko hipervizorja, ampak Windows Sandbox obravnava peskovnik skoraj kot aplikacija na gostitelju, zaradi česa lahko gostitelj načrtuje izvajanje svojih procesov in procesov peskovnika.[9][10]

5.3 Virtualizacija grafičnih naprav

Uporaba grafičnih naprav za pospeševanje grafik v navideznih strojih je težavno. Deli operacijskega sistema Windows, ki skrbijo za grafiko, so ustvarjeni tako, da se zavedajo stanja navideznih strojev, in se lahko pogovarjajo med gostiteljem in navideznim strojem.[9][10]

Aplikacija v navideznem stroju, ki zahteva klic na grafični vmesnik, to naredi, kot če ne bi bila virtualizirana, saj se samodejno posreduje naprej do grafične naprave preko meje med navideznim strojem in gostiteljem.[9][10]

Poglavje 6

Sklepne ugotovitve

Peskovnik je zelo uporabno orodje za izboljševanje varnost računalniških sistemov. Omejitev kode in kako dostopa do podatkov je ključno za onemogočanje zlorabe.

Peskovnikov je zelo veliko in dandanes so prisotni skoraj vsepovsod, vse od operacijskih sistemov, kot so Windows, ki lahko vedno tečejo v navideznem stroju s pomočjo Hyper-V virtualizacije, do kompleksnih peskovnikov v naših brskalnikih in seveda v praktično vseh pametnih telefonih z Android Application Sandbox.

Literatura

[1] Dosegljivo: https://source.android.com/security/overview/kernel-security, . [Dostopano: 8.1.2022].

[2] Dosegljivo: https://source.android.com/security/app-sandbox, . [Dostopano: 8.1.2022].

[3] Dosegljivo: https://chromium.googlesource.com/chromium/src/+/refs/heads/main/docs/design/sandbox_faq.md, . [Dostopano: 8.1.2022].

[4] Dosegljivo: https://chromium.googlesource.com/chromium/src/+/refs/heads/main/docs/design/sandbox.md, . [Dostopano: 8.1.2022].

[5] Dosegljivo: https://docs.freebsd.org/en/books/developers-handbook/secure/#secure-chroot, . [Dostopano: 8.1.2022].

[6] Dosegljivo: https://en.wikipedia.org/wiki/FreeBSD_jail, . [Dostopano: 8.1.2022].

[7] Dosegljivo: https://en.wikipedia.org/wiki/Seccomp. [Dostopano: 8.1.2022].

[8] Dosegljivo: https://en.wikipedia.org/wiki/Virtualization. [Dostopano: 8.1.2022].

[9] Dosegljivo: https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-sandbox/windows-sandbox-architecture, . [Dostopano: 8.1.2022].

[10] Dosegljivo: https://techcommunity.microsoft.com/t5/windows-kernel-internals-blog/windows-sandbox/ba-p/301849, . [Dostopano: 8.1.2022].