Sistemski poziv ptrace i njegova uloga u radu debagera.

Seminarski rad u okviru kursa
Verifikacija softvera
Matematički fakultet

Nikola Dimitrijević, 1086/2017
nikoladim95@gmail.com

11. novembar 2018

Sažetak

Ptrace je sistemski poziv u Unix i Unixolikim operativnim sistemima. Ime je skraćenica od "proces tragač" (eng. *proecss trace*). Korišćenjem ptrace jedan proces može da kontroliše drugi. Time upravljački proces ima mogućnost da upravlja unutrašnjim stanjem ciljanog procesa. Debageri najviše koriste ptrace kako bi mogli da zaustavljaju i posmatraju memoriju programa.

Sadržaj

1 Uvod **2**

2 Upotrebe **2**

3 Povezivanje **3**
3.1 Ispod haube 3
3.2 Povratna vrednost 4
3.3 Bezbednost 4

4 Slike i tabele **4**
4.1 Prvi podnaslov 4

Literatura **4**

1 Uvod

Sistemski poziv *ptrace* pruža mogućnost da jedan proces, posmatra i kontroliše izvršavanje drugog procesa kao i da posmatra i menja memoriju i registre traženog procesa. Za proces koji upravlja drugim se koristi termin tragač (eng. *tracer*), a proces kojim on upravlja se naziva traženi (eng. *tracee*) proces.

Koristi se za implementaciju debagera i praćenja sistemskih poziva.

Prvo je potrebno da traženi proces bude povezan na tragača. Povezivanje i sve prateće komande se zasebno rade po niti procesa. Ptrace komande se uvek šalju određenom traženom procesu u sledećem formatu:

```
ptrace(enum zahtev, pid_t proces, void* adresa, void* podaci)
```

Argumenti adresa i podaci se koriste ili ignorisu u zavisnosti od vrste zahteva.

2 Upotrebe

Debageri poput *gdb* koriste ptrace, alati *ltrace* i *strace* i alati za računanje test pokrivenosti koda. Povezivanjem na drugi proces korišćenjem ptrace roditeljski proces dobija veliku kontrolu nad ciljanim procesom. Tu spadaju:

- desktriptori datoteka
- memorija
- registri
- izvršavanje instrukciju po instrukciju
- posmatranje i presretanje sistemskih poziva
- uvid u povratne vrednosti sistemskih poziva
- manipulisanje upravljacem signala
- primanje i slanje signala umesto procesa

Menjanjem memorije programa se, pored menjanja podataka programa, može menjati i segmenat koda. Na taj način kontroler proces može da ubaci brejkpointe (eng. *brakepoints*) i da izmeni instrukcije programa prilikom njegovog izvršavanja.

Pomenuti alat ltrace služi za prikaz poziva koje program salje deljenim bibliotekama. To radi tako što se zakači na sistem za dinamičko punjenje pa tako može da vidi funkcije, parametre i povratne vrednosti poziva biblioteka [4].

Alat strace se koristi za debagovanje i dijagnostiku na Linuks operativnom sistemu. Može da vidi i barata sa interakcijama izmešu procesa i Linuks jezgra, što uključuje sistemske pozive, dostavljanja signala i menjanje stanja procesa [6].

Oba alata pokreću zadati program do kraja ivršavanja.

Debagerima je ptrace poput švajcarskog nožića, ali kako zapravo debageri postavljaju brejkpointe? Oni zamene instrukciju koja bi trebalo da se izvrši sa instrukcijom zamke (eng. *trap instruction*) [2]. Kada se traženi proces zaustavi onda program tragač, debager, može da ga proučava. Kada je potrebno da traženi program nastavi izvršavanje onda će se umesto brejkpointa vratiti prvobitna instrukcija na svoje mesto.

3 Povezivanje

Vaš seminarski rad mora da sadrži najmanje četiri reference u spisku literature. **Dužina seminarskog rada treba da bude najmanje 4 strane teksta.**

Tragač može da se poveže na nit korišćenjem poziva:

```
ptrace(PTRACE_ATTACH, pid, 0, 0);
```

ili

```
ptrace(PTRACE_SEIZE, pid, 0, PTRACE_O_FLAGS);
```

PTRACE_ATTACH salje signal SIGSTOP ciljanoj niti [5].

Novije verzije Linux-a imaju opciju

PTRACE_SEIZE umesto PTRACE_ATTACH.

PTRACE_SEIZE ne zaustavlja prikačeni proces.

Ako je potrebno zaustavljanje nakon priključenja može se pozvati ptrace sa opcijom PTRACE_INTERRUPT.

Jedini ptrace poziv koji traženi proces koristi je PTRACE_TRACEME, sve ostale pozive koristi samo tragač. PTRACE_TRACEME govori roditelju procesu da želi da bude praćen.

Dobra praksa je da se nakon PTRACE_TRACEME pozove

```
raise(SIGSTOP);
```

i tako dopusti roditeljskom procesu, koji je sada tragač, da posmatra dostavljanje signala. Dakle roditeljski proces može da započne traganje, ali i dete proces može da to zatraži od roditelja. Ne bi trebalo da dete proces pozive PTRACE_TRACEME ako roditeljski proces to ne očekuje. Nakon pozivanja TRACEME program nastavlja sa izvršavanjem, ali svaki sledeći signal dostavljen tom procesu će uslediti njegovom zaustavljanju i njegov roditeljski proces će biti obavešten putem wait(). Jedino signal SIGKILL neće biti prvo prosleđen roditelju nego će se proces zaustaviti svakako.

Za određeni proces samo jedan proces može da mu bude tragač u bilo kom trenutku.

U slučaju da je proces ptraceovan pomoću PTRACE_ATTACH onda traženi proces nastavlja sa izvršavanjem sve dok ne uđe u sistemski poziv, u tom trenutku ga zaustavlja Linux jezgro. Traženom procesu to izgleda kao da je zaustavljen jer je primio SIGTRAP signal. Sada tragač može da radi šta mu je volja.

3.1 Ispod haube

Kako je kod javno dostupan [3], može se pogledati implementacija poziva ptrace.

```
if (request == PTRACE_ATTACH || request == PTRACE_SEIZE) {
        ret = ptrace_attach(child, request, addr, data);
        /*
        * Some architectures need to do book-keeping after
        * a ptrace attach.
        */
        if (!ret)
                arch_ptrace_attach(child);
        goto out_put_task_struct;
}
```

Nakon provere da li je zahtev PTRACE_ATTACH es poziva ta funkcija. Ona prvo postavlja oznake koje će posle biti zabeležene u jezgru kao reprezentacija traženog procesa. Proverava da traženi zadatak (eng. *task*) nije nit jezgra. Takođe proverava da se ne pokušava povezivanje procesa sa samim sobom. Sada je traženi proces zaustavljen. Na kraju ptrace zove arch_ptrace, funkciju koja zavisi od procesora, tako da je implementaicja usko vezana za konkretnu arhitekturu.

Postavljena je oznaka da je proces tražen, sad je pitanje kada se ta oznaka proverava prilikom sistemskih poziva. Svaki put kada program napravi sistemski poziv, postoji kod zavisan od arhitekture procesora koji se izvrši na strani jezgra pre nego što se izvrši sistemski poziv.

U x86 asemblerskoj implementaciji se vidi provera pomenute postavljene oznake[1]. Tako se vidi da se pri svakom sistemskom pozivu vrši ta provera. U slučaju da je oznaka postavljena poziva se nekoliko funkcija i konačno se okida signal SIGTRAP. Tragač je tada obavešten o primljenom signalu, a traženi proces je zaustavljen. Sada je tragač u stanju da ispituje unutrašnjost traženog procesa.

3.2 Povratna vrednost

Pri uspehu, PTRACE_PEEK zahtevi vraćaju tražene podatke, dok ostali zahtevi vraćaju nulu. Svi zahtevi vraćaju -1 pri neuspehu i *errno* bude postavljen na odgovarajuću vrednost. Tip povratne vrednosti je long.

3.3 Bezbednost

Operativni sistemi pružaju usluge programima preko standardnog API-ja za pristup hardveru i drugi sistemima niskog nivoa poput sistema datoteka. Kada proces hoće da zove sistemski poziv prvo postavlja svoje argumente u registre i zove softverski prekid. Ovaj prekid govori jezgru da proveri argumente i potom da izvrši sistemski poziv.

Dinamičko ubacivanje koda se koristi za omogućavanje debagovanja, ali se isto može koristiti i za zlonamerne aktivnosti ako napadač ima privilegije da pokrene ptrace.

4 Slike i tabele

Slike i tabele treba da budu u svom okruženju, sa odgovarajućim naslovima, obeležene labelom da koje omogućava referenciranje.

Primer 4.1 *Ovako se ubacuje slika. Obratiti pažnju da je dodato i*

`\usepackage{graphicx}`

Na svaku sliku neophodno je referisati se negde u tekstu. Na primer, na slici 1 prikazane su pande.

4.1 Prvi podnaslov

Literatura

[1] How does strace work?

[2] Linux journal. https://www.linuxjournal.com/article/6210.

Slika 1: Pande

Tabela 1: Razlčita poravnanja u okviru iste tabele ne treba koristiti jer su nepregledna.

centralno poravnanje	levo poravnanje	desno poravnanje
a	b	c
d	e	f

[3] linux/kernel/ptrace.c.

[4] ltrace man page.

[5] ptrace man page.

[6] strace man page.