LINUX RENDSZERFEJLESZTÉS ÉS ÜZEMELTETÉS

2. forduló

ONE IDENTITY by Quest

A kategória támogatója: One Identity

RENDELKEZÉSRE ÁLLÓ IDŐ:

60:00

Ismertető a feladathoz

Fontos információk

> Ha kifutsz az adott feladatlap kitöltésére rendelkezésre álló időből, a felület **automatikusan megpróbálja beküldeni** az addig megadott válaszokat
>
> A kérdésekre **mindig van helyes válasz**! Ha csak egy helyes válasz van az adott kérdésre, radio button-os választási lehetőségeket fogsz látni.
>
> **Olyan kérdés viszont nincs, amelyre az összes válasz helyes!**

Egyéb információkat a versenyszabályzatban találsz!

Második forduló

Programozzunk!

Felhasznált idő: 01:44/60:00 **Elért pontszám:** 0/10

1. feladat 0/10 pont

Az egyik linux üzemeltető a következő üzenetet látja az egyik linux szerver root konzolján: "bash: fork: retry: Resource temporarily unavailable", valamint nem kap promptot már régóta. Saját elmondása szerint minimum fél órája. Hosszas tanakodás után arra jutottunk, hogy valamiféle fork bomba szabadult el a gépen. Sajnos a szerveren egy adatbázisszerver is működik, ezért a reset csak az utolsó esély, mert nagyobb baj is lehet abból, hogy a memóriában rengetek adat van cache-elve, ami nem biztos, hogy elért a diskig, amikor reseteljük a szervert. Míg tanakodtunk, az egyik óvatlan diák a saját konzolján kiadott egy "ls"-t a /proc könyvtár alatt, amiből az látszik, hogy rengeteg könyvtár van a /proc könyvtárban. Egy "cat" paranccsal viszont már nem ment sokra -- hogy belenézzen egy cmdline tartalmába --, mert az már nem futott le. Ez megerősíti a gyanúnkat. A szerver egy ubuntu 16.04-es gép 4.4-es kernellel, ahol a pid cgroup be van kapcsolva, és a gép

50GB memóriával rendelkezik. Szerencsére nincs minden veszve, ugyanis egy root ssh konzol még nyitva van a szerverhez. Annyit tudunk még az "ls" parancs kimenetéből, hogy a fork bomba pidje valahol 1500 felett indult, ami magasabb a futó adatbázis pidjeinél. Milyen parancsot vagy parancsokat adjunk ki, hogy -- ha szerencsénk van -- visszaszerezhessük az irányítást?

Válasz

- `killall -STOP -u root`
- `killall -KILL -u root`
- `for i in $(seq 1000 999999); do /bin/kill -9 $i; done`
- `echo "root hard nproc 1500" >> /etc/security/limits.conf`
- `i=1500; while true; do let "i=i+1"; kill -9 $i; done`
- `csak várni kell amíg elmúlik a fork bomba, hála a pids cgroup-nak`

Magyarázat

Mivel fork bombáról van szó, naiv gondolat arra számítani, hogy bármilyen futtatható programot el fogunk tudni indítani, de a bash-nek van egy kill belső parancsa. A kill parancs a "/bin"-ben nem a belső parancs. Emiatt az első 3 megoldás nem jó. A negyedik azért nem jó, mert azzal még nem fogjuk tudni visszaszerezni az irányítást, mivel userként talán tudunk majd parancsokat kiadni, de rootként semmiképpen. A 6. válasz meg a megadott OS kernel párosítás miatt nem működik. Megjegyzem, hogy a következő LTS-ben már működni fog a cgroup ezen funkciója.

2. feladat 0/0 pont

Egy ismeretlen számrendszerbeli m42a8221 számot egy másik ismeretlen számrendszerbe átszámolva, majd abból egy számot levonva a szintén ismeretlen számrendszerbeli 21mpd95h számot kapjuk eredményül. Azt tudjuk még, hogy a levonandó szám számjegyei balról-jobbra egy számtani sor elemei. Az egyszerűség kedvéért tegyük fel, hogy a feladatban szereplő számrendszerek egész számú számrendszerek. Mi a levonandó szám 12-es számrendszerben ?

Válaszok

A helyes válasz:
929a6a3216
929A6A3216

Magyarázat

A feladatot a sok versenyzői visszajelzést megfontolva 0 pontosra állítottuk. Annak a pár játékosnak, akiknek sikerült megoldaniuk, gratulálunk! (Azért nem a törlés mellett döntöttünk, mert többen jeleztétek, hogy a megoldókucs ismeretében kipróbálnátok a megoldást.)

A rengeteg lehetséges megoldás közül csak egy program segítségével tudjuk kiválasztani a helyes választ, amire minden feltétel teljesül. Ilyen program például:

```
import sys

def search(a, b, min_base = 2, max_base = 36):
    for source_base in range( min_base, max_base + 1 ):
        try:
            num_a = int( a, source_base )
        except ValueError:
            continue

        for target_base in range( min_base, max_base + 1 ):
            try:
                num_b = int( b, target_base )
            except ValueError:
                continue

            for diff_base in range( min_base, max_base + 1 ):
                diff = int_to_base( num_a - num_b, diff_base )
                # print("try:", int_to_base( num_a, source_base ), source_base, int_to_base( num_b, ta

                if sequence_is_valid( diff, diff_base ):
                    print("result:", int_to_base( num_a, source_base ), source_base, int_to_base( num_
    return 0

def sequence_is_valid(n, base):
    if n[ 0 ] == "-":
        n = n[ 1: ]

    sequence = int( str( n )[ 1 ], base ) - int( str( n )[ 0 ], base )
    sequence_is_validity = True

    for digits in range( 2, len( str( n ) ) ):
        if int( str( n )[ digits ], base ) - int( str( n )[ digits - 1 ], base ) != sequence:
            sequence_is_validity = False
            break

    return sequence_is_validity

def int_to_base(n, base, digits="0123456789abcdefghijklmnopqrstuvwxyz"):
    sgn = ""
```

```
    if n < 0:
        sgn = "-"
        n = -n

    return sgn + ( int_to_base( n // base, base, digits ) if n >= base else "" ) + digits[ n % base ]

if __name__ == "__main__":
    source_base = 23
    target_base = 28
    diff_base = 34

    a    = "m42a8221"
    diff =  "usqomki"
    b    = int_to_base( int( a, source_base ) - int( diff, diff_base ), target_base )

    print("task:", a, source_base, b, target_base, diff, diff_base)
    search( a, b )
    print("diff in base 12:", int_to_base( int( diff, diff_base ), 12 ) )
```