Systemy operacyjne

Lista zadań nr 4

Na zajęcia 5 listopada 2020

Należy przygotować się do zajęć czytając następujące rozdziały książek:

- Arpaci-Dusseau: 39 (Files and Directories[1])
- Tanenbaum (wydanie czwarte): 4.1, 4.2, 10.6
- APUE (wydanie trzecie): 3, 4, 5
- Linux Programming Interface: 4, 5, 18, 44

UWAGA! W trakcie prezentacji należy być gotowym do zdefiniowania pojęć oznaczonych **wytłuszczoną** czcionką.

Zadanie 1. Czemu wywołania `read(2)` i `write(2)` nie działają na katalogach? Jakim wywołaniem systemowym można wczytać **rekord katalogu** (ang. *directory entry*)? Czy zawartość katalogu jest posortowana? Wyświetl **metadane** katalogu głównego «/» przy pomocy polecenia «`stat`», a następnie wyjaśnij z czego wynika podana liczba **dowiązań** (ang. *hard link*)?

Zadanie 2. Rura `pipe(7)` to **jednokierunkowe** narzędzie do komunikacji międzyprocesowej. Co robi operacja `read(2)` i `write(2)`, jeśli **bufor** rury jest odpowiednio pusty albo pełny? Rozważamy wiele procesów piszących do tej samej rury wiersze tekstu nie dłuższe niż «`PIPE_BUF`» – co to znaczy, że zapisy są atomowe? Co się stanie, jeśli zostanie zamknięty jeden z końców rury? Kiedy operacje `read` i `write` na rurze zwracają „short count”? Czy można połączyć rodzica i dziecko rurą, która została utworzona po uruchomieniu dziecka?

Zadanie 3. Uruchamiamy w powłoce **potok** (ang. *pipeline*) «`ps -ef | grep sh | wc -l > cnt`». Po zakończeniu działania polecenia do powłoki zostanie przekazany kod wyjścia ostatniego procesu w potoku. Uzasadnij kolejność tworzenia procesów potoku posługując się obrazem 9.10 z rozdziału „Shell Execution of Programs” (APUE). Wskaż które procesy i w jakiej kolejności będą wołały: `creat(2)`, `dup2(2)`, `pipe(2)`, `close(2)`, `waitpid(2)`, `fork(2)`, `execve(2)`. Co jeśli jedno z wywołań «`execve`» zawiedzie?

Zadanie 4. Zapoznaj się z krytyką interfejsu plików przedstawioną w podrozdziale „*`ioctl` and `fcntl` Are an Embarrassment*”[2]. Do czego służy wywołanie systemowe `ioctl(2)`? Zauważ, że stosowane jest głównie do plików **urządzeń znakowych** lub **blokowych**. Na podstawie pliku `ioccom.h`[3] wyjaśnij znaczenie drugiego i trzeciego parametru wywołania `ioctl`. Używając przeglądarki kodu[4] jądra NetBSD znajdź definicje operacji «`DIOCEJECT`», «`KIOCTYPE`» i «`SIOCGIFCONF`», a następnie wytłumacz co one robią.

Komentarz: Prowadzący przedmiot zgadza się z autorem krytyki. Czy i Ty widzisz brzydotę tego interfejsu?

[1] `http://pages.cs.wisc.edu/~remzi/OSTEP/file-intro.pdf`
[2] `http://www.catb.org/~esr/writings/taoup/html/ch20s03.html`
[3] `https://nxr.netbsd.org/xref/src/sys/sys/ioccom.h`
[4] `https://nxr.netbsd.org`

Zadanie 5. Intencją autora poniższego kodu było użycie plików jako blokad międzyprocesowych. Istnienie pliku o podanej nazwie w systemie plików oznacza, że blokada została założona. Brak tegoż pliku, że blokadę można założyć. Niestety w poniższym kodzie jest błąd TOCTTOU[5], który opisano również w §39.17. Zlokalizuj w poniższym kodzie wyścig i napraw go! Opowiedz jakie zagrożenia niesie ze sobą taki błąd.

```
1  #include "csapp.h"
2
3  bool f_lock(const char *path) {
4    if (access(path, F_OK) == 0)
5      return false;
6    (void)Open(path, O_CREAT|O_WRONLY, 0700);
7    return true;
8  }
9
10 void f_unlock(const char *path) {
11   Unlink(path);
12 }
```

Wskazówka: Przeczytaj komentarze do flagi «O_CREAT» w podręczniku do open(2).

Ściągnij ze strony przedmiotu archiwum «so20_lista_4.tar.gz», następnie rozpakuj i zapoznaj się z dostarczonymi plikami.
UWAGA! Można modyfikować tylko te fragmenty programów, które zostały oznaczone w komentarzu napisem «TODO».

Zadanie 6. Program «leaky» symuluje aplikację, która posiada dostęp do danych wrażliwych. Pod deskryptorem pliku o nieustalonym numerze kryje się otwarty plik «mypasswd». W wyniku normalnego działania «leaky» uruchamia zewnętrzny program «innocent» dostarczony przez złośliwego użytkownika.

Uzupełnij kod programu «innocent», aby przeszukał otwarte deskryptory plików, a następnie przepisał zawartość otwartych plików do pliku «/tmp/hacker». Zauważ, że pliki zwykłe posiadają **kursor**. Do pliku wyjściowego należy wpisać również numer deskryptora pliku i ścieżkę do pliku, tak jak na poniższym wydruku:

```
1 File descriptor 826 is '/home/cahir/lista_4/mypasswd' file!
2 cahir:...:0:0:Krystian Baclawski:/home/cahir:/bin/bash
```

Żeby odnaleźć nazwę pliku należy wykorzystać zawartość katalogu «/proc/self/fd» opisaną w procfs(5). Potrzebujesz odczytać plik docelowy odpowiedniego **dowiązania symbolicznego** przy pomocy readlink(2).

Następnie napraw program «leaky» – zakładamy, że nie może on zamknąć pliku z wrażliwymi danymi. Wykorzystaj fcntl(2) do ustawienia odpowiedniej flagi deskryptora wymienionej w open(2).

Zainstaluj pakiet «john» (John The Ripper[6]). Następnie złam hasło znajdujące się pliku, który wyciekł w wyniku podatności pozostawionej przez programistę, który nie przeczytał uważnie podręcznika do execve(2).

Wskazówka: Procedura «dprintf» drukuje korzystając z deskryptora pliku, a nie struktury «FILE».

Zadanie 7. (Pomysłodawcą zadania jest Tomasz Wierzbicki.)

Program «primes» używa Sita Eratostenesa[7] do obliczania liczb pierwszych z przedziału od 2 do 10000. Proces główny tworzy dwóch potomków wykonujących procedurę «generator» i «filter_chain», spiętych rurą «gen_pipe». Pierwszy podproces wpisuje do rury kolejne liczby z zadanego przedziału. Drugi podproces tworzy łańcuch procesów filtrów, z których każdy jest spięty rurą ze swoim poprzednikiem. Procesy w łańcuchu powstają w wyniku obliczania kolejnych liczb pierwszych. Każdy nowy filtr najpierw wczytuje liczbę pierwszą p od poprzednika, po czym drukuje ją, a następnie kopiuje kolejne liczby z poprzednika do następnika za wyjątkiem liczb podzielnych przez p.

Program musi poprawnie działać dla argumentu 10000 – w tym przypadku powinno zostać utworzonych $1229 + 2$ podprocesów.

Uwaga! Rozwiązania, które nie zapewniają pochówku umarłym dzieciom lub nie dbają o zamykanie nieużywanych końców rur, są uważane za błędne. Będziemy to sprawdzać poleceniem «ps» i «lsof».

[5] https://www.usenix.org/legacy/event/fast05/tech/full_papers/wei/wei.pdf
[6] https://www.openwall.com/john/
[7] https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes