Systemy operacyjne

Lista zadań nr 4

Na zajęcia 5 i 13 listopada 2019

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

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

Zadanie 1. Wywołanie «execve» częściowo zastępuje **obraz procesu** (ang. *process image*). Na podstawie podręcznika `execve(2)` wyjaśnij, które zasoby są dziedziczone przez świeżo załadowany program. Co dzieje się z otwartymi plikami i obsługą sygnałów? Co robią programy `passwd(1)` i `wall(1)`? Mają one ustawione **bity uprawnień** odpowiednio «set-uid» i «set-gid». Uzasadnij, że jest to konieczne do ich działania.

Zadanie 2 (bonus). Wywołanie `open(2)` zawiedzie z błędem «ETXTBSY», jeśli próbujemy otworzyć plik wykonywalny do zapisu pod warunkiem, że tenże plik jest w chwili obecnej wykonywany. Co mogłoby się stać, gdyby system operacyjny pozwolił modyfikować plik wykonywalny, który jest uruchomiony?

Wskazówka: Rozważamy system ze stronicowaniem na żądanie (ang. *demand paging*).

Zadanie 3. W systemach uniksowych wszystkie pliki są ciągiem bajtów. Wyjątkiem są katalogi będące listą rekordów «dirent» opisanych w `readdir(3)`. Które z operacji wymienionych w §39 działają na katalogach? Na podstawie §10.6.3 (rysunek 10-32) przedstaw reprezentację katalogu i pokaż jak przebiega operacja usuwania pliku. Wpis katalogu nie zawiera **metadanych** pliku – gdzie w takim razie są one składowane? Co koduje pole «st_link» struktury «statbuf» opisanej w `stat(2)`? Kiedy plik zostaje faktycznie usunięty z dysku? Zauważ, że nadal możesz czytać z otwartego pliku, który został usunięty!

Zadanie 4 (P). Jaką rolę pełnią bity uprawnień «rwx» dla katalogów w systemach uniksowych? Opisz znaczenie bitów «set-gid» i «sticky» dla katalogów. Zaprezentuj pseudokod procedury «`bool my_access(struct stat *statbuf, int mode)`». Pierwszy argument opisano w `stat(2)`, a drugi w `access(2)`. Dla procesu o tożsamości zadanej przez `getuid(2)` i `getgroups(2)` procedura «my_access» odpowiada czy proces ma dostęp «mode» do pliku o metadanych wczytanych do «statbuf».

Wskazówka: Rozważ uprawnienia katalogu «/usr/local» i «/tmp».

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[2], 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)`.

[1] http://pages.cs.wisc.edu/~remzi/OSTEP/file-intro.pdf
[2] https://www.usenix.org/legacy/event/fast05/tech/full_papers/wei/wei.pdf

Ściągnij ze strony przedmiotu archiwum «so19_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 (P). 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[3]). 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 (P). Uruchom program «mkholes», a następnie odczytaj **metadane** pliku «holes.bin» przy pomocy polecenia stat(1). Wszystkie pola struktury «stat» są opisane w stat(2). Oblicz faktyczną objętość pliku na podstawie liczby używanych bloków «st_blocks» i rozmiaru pojedynczego bloku «st_blksize» systemu pliku. Czemu liczba używanych bloków jest mniejsza od tej wynikającej z objętości pliku z pola «st_size»? Czemu jest większa od liczby faktycznie używanych bloków zgłaszanych przez «mkholes»?

Wskazówka: O dziurach w plikach (ang. *holes*) można przeczytać w rozdziale 3.6 APUE.

Zadanie 8 (P). Uzupełnij program «game» tj. prostą grę w szybkie obliczanie sumy dwóch liczb. Zadaniem procedury «readnum» jest wczytać od użytkownika liczbę. Jeśli w międzyczasie przyjdzie sygnał, to procedura ma natychmiast wrócić podając numer sygnału, który przerwał jej działanie. W przeciwnym przypadku zwraca zero i przekazuje wczytaną liczbę przez pamięć pod wskaźnikiem «num_p». Twoja implementacja procedury «readnum» musi wczytać całą linię w jednym kroku! Należy wykorzystać procedury siglongjmp(3), sigsetjmp(3) i alarm(2). Kiedy Twój program będzie zachowywać się poprawnie zamień procedury **nielokalnych skoków** na longjmp(3) i setjmp(3). Czemu program przestał działać?

UWAGA! We FreeBSD i macOS zamiast «longjmp» i «setjmp» należy użyć odpowiednio «_longjmp» i «_setjmp».

[3] https://www.openwall.com/john/