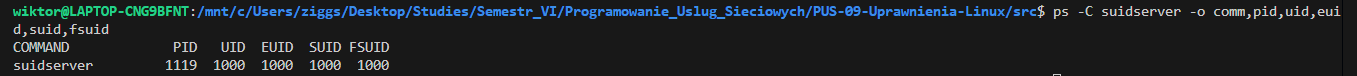
**Uprawnienia Linux  
Laboratorium 9  
Wiktor Zmiendak**

**ZADANIE 1**

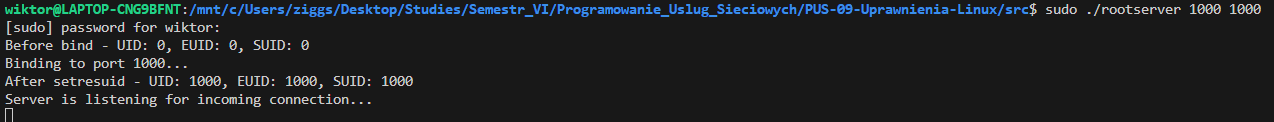
**Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.**

****

Program suidserver to prosty serwer TCP, który może nasłuchiwać na portach uprzywilejowanych (<1024) dzięki mechanizmowi SUID. Po skompilowaniu i ustawieniu odpowiednich uprawnień, możliwe jest jego uruchomienie przez zwykłego użytkownika. Program tworzy gniazdo, wiąże je z portem, a następnie zrzuca przywileje roota, kontynuując działanie jako użytkownik nieuprzywilejowany. Po nawiązaniu jednego połączenia przez klienta, program kończy działanie. Przy uruchomieniu z konta roota, program wypisze identyfikatory UID, EUID i SUID jako równe 0, ale samodzielnie zrzuca uprawnienia przez setuid(getuid()), by zapobiec dalszemu działaniu z prawami administratora.

**ZADANIE 2**

****

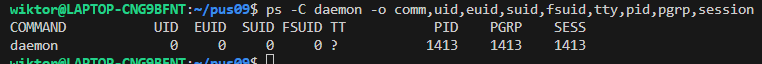
**Obraz zawierający tekst, zrzut ekranu, Czcionka

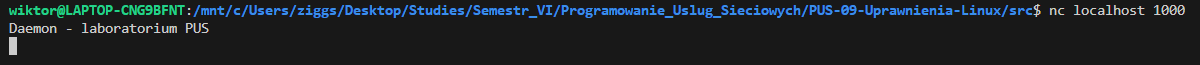
Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.**

Zmieniona wersja programu rootserver uruchamiana jest z konta roota i pozwala nasłuchiwać na uprzywilejowanym porcie (<1024). Po wykonaniu operacji wymagających uprawnień administratora, program zmienia swoje identyfikatory użytkownika (UID, EUID, SUID) na wskazane w argumencie wywołania. W ten sposób dalsze działanie programu odbywa się w kontekście zwykłego użytkownika. Po uruchomieniu serwera i sprawdzeniu procesu przy użyciu ps, można zaobserwować, że wszystkie identyfikatory UID zostały poprawnie zmienione.

**ZADANIE 3**

****

****

****

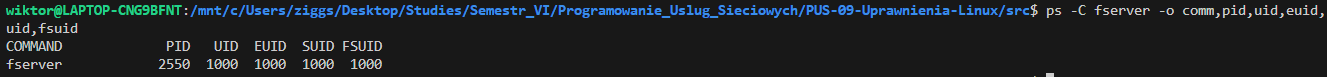
Program tworzy serwer TCP nasłuchujący na porcie (>1024), który po uruchomieniu demonizuje się funkcją daemonize().  
Funkcja ta wykonuje standardowe kroki demonizacji: tworzy proces potomny (fork), który staje się liderem nowej sesji (setsid), zmienia katalog roboczy na / oraz przekierowuje standardowe wejście, wyjście i błędy na /dev/null. Dzięki temu proces działa w tle, niezależnie od terminala.

Po demonizacji proces działa jako daemon — jego PID jest różny od terminala, jest liderem własnej grupy procesów i sesji (PID = PGID = SID), a powiązanie z terminalem (TTY) jest zerwane (brak terminala).

Test połączenia przez nc localhost <port> potwierdza poprawne działanie serwera, który po przyjęciu połączenia wysyła komunikat i zamyka połączenie.

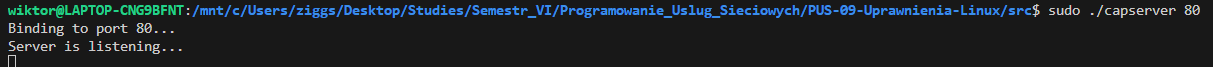
**ZADANIE 4**

****

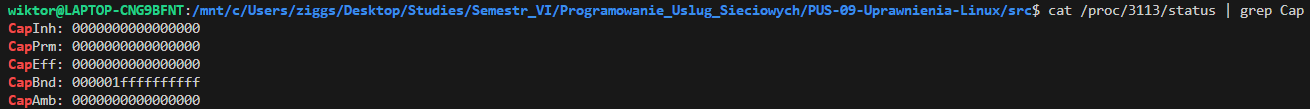
****

Celem było uruchomienie serwera TCP nasłuchującego na porcie < 1024 bez konieczności uruchamiania programu jako root. Usunięto z kodu wywołania setuid() oraz getresuid(). Skonfigurowano program w sposób umożliwiający wykorzystanie uprawnienia CAP\_NET\_BIND\_SERVICE. Dzięki nadaniu uprawnienia CAP\_NET\_BIND\_SERVICE, możliwe było powiązanie gniazda z portem < 1024 bez potrzeby nadawania uprawnień roota ani stosowania setuid(). Jest to bezpieczniejsza alternatywa, pozwalająca ograniczyć zakres uprawnień wykonywanego programu.

**ZADANIE 5**

****

****

****

Celem było uruchomienie serwera TCP nasłuchującego na porcie < 1024 bez przyznawania mu efektywnych uprawnień CAP\_NET\_BIND\_SERVICE, a jedynie *permitted*. Po wykonaniu operacji bind(), program zrzuca wszystkie uprawnienia przy pomocy biblioteki libcap. Dzięki wykorzystaniu tylko zbioru permitted, ograniczono uprawnienia programu do minimum, co zwiększa bezpieczeństwo. Dodatkowo, po wykonaniu bind(), zrzucenie capabilities uniemożliwia dalsze wykonywanie operacji wymagających podwyższonych uprawnień.

**ZADANIE 6**