Zadanie 5.

Zasada najbardziej ograniczonych uprawnień mówi, że ktoś, kto działa (użytkownik, program), powinien mieć uprawnienia tylko do tego, czego potrzebuje.

Capsicum jest frameworkiem, który pozwala na inne podejście do uprawnień. Zamiast podejścia, w którym mamy uprawnienia naszego właściciela i możemy robić wszystko to, co on by mógł. Mamy *capabilities*, tzn. dostajemy pozwolenie na wykonanie konkretnych akcji. Np. dostajemy dostęp do jednego folderu i możemy sobie w nim działać (za pomocą openat itp.), ale nie możemy wyjść poza niego.

Capsicum (w przeciwieństwie do chroot i efemerycznych UIDów) nie wymaga uprzywilejowania.

Kod, który ma dostęp do danych (parsowanie danych i kompresja) jest zagrożony na wykorzystanie luk. Jednocześnie gzip musi móc utworzyć plik w dowolnym miejscu i czytać pliki z dowolnego miejsca (tzn. takie, do których właściciel ma dostęp). Dlatego gzip został podzielony tak, że główna część, która wczytuje argumenty, otwiera pliki na wejście i wyjście, a następnie przekazuje deskryptory plików (każdy z odpowiednią kombinacją CAP_READ, CAP_WRITE, CAP_SEEK) do metod kompresji.

Podproces przeprowadzający kompresję daych powinien mieć uprawnienia do: * CAP_SEEK, CAP_READ - na plikach wejściowych i wyjściowych * CAP_WRITE - na pliku wynikowym

(Zakładam, że on nie musi otwierać plików, bo dostał na nie fd od głównego procesu).

Zadanie 6.

unveil pozwala na ograniczenie uprawnień do systemu plików. Pierwsze wywołanie sprawia, że inne wywołania systemowe działające na systemie plików (open, chmod itd.) nie widzą nic (unveil widzi). Jako argumenty podajemy ścieżkę i uprawnienia, w ten sposób proces nabywa uprawnienia do wszystkiego pod tą ścieżką (najniższy przodek w drzewie systemu plików na największy prioritet). Wołając *unveil(NULL, NULL)*, tracimy dostęp do wywołania unveil.

chroot ustawia root directory procesu na daną ścieżkę. To tylko zmienia sposób rozwiązywania ścieżek i nie służy do sandboxowania.

Przeglądarka musi mieć dostęp do niektórych plików np. czcionek. Za pomocą unveil można ograniczyć jej dostęp do sytemu plików w taki sposób, żeby miała dostęp tylko do tego, co konieczne.

Zadanie 7.

prawo Amdahla – czas działania, skrócony zrównolegleniem wykonywania operacji jest ograniczony przez czas działania tych części programu, których nie da się zrównoleglić.

Nie da się zrównoleglić sortowania przez scalanie oraz wybierania pivota (ale to jest w czasie stałym). Dzielenie ciągu na dwie części da się zrównoleglić (ale to by wymagało komunikacji pomiędzy procesami, które przeglądają poszczególne cześci ciągu elementów), więc podejrzewam, że tutaj też chodzi o to, że nie.

zużycie procesora – łączny czas, przez który proces i jego potomkowie zajęli na CPU (user+sys). **turnaround time** – czas przebywania w systemie, czas od utworzenia do zakończenia procesu (real).

Przed i po zrównolegleniu

```
real    0m7.793s
user    0m7.545s
sys 0m0.227s

real    0m2.736s
user    0m12.371s
sys 0m0.583s
```