Paweł Śnioszek - 145245

Sprawozdanie sk2
Zdalne zamykanie systemów operacyjnych

1. Opis projektu

- Zaimplementowana aplikacja służy do zdalnego zamykania systemów operacyjnych. Aplikacja zrealizowana jest w architekturze klient-serwer. Główny serwer zarządza całością działania aplikacji. To do niego podłączają się klienci oraz agenci. Każde nowe, zaakceptowane połączenie tworzy nowy wątek do obsługi komunikacji. Istnieją jednak ograniczenia, ponieważ zabronione jest podłączenie się z jednej maszyny (tego samego IP) więcej niż jednego agenta. Każdy zaakceptowany agent musi podać hasło. Jest ono używane do weryfikacji uprawnień klienta. Tylko klient ze znajomością hasła danego agenta może wyłączyć maszynę na której ten pracuje. Przyznane klientowi uprawnienia do danej maszyny zapamiętywane są w ramach jednej sesji. Identyfikatorem danej maszyny jest jej adres IP.

- Do zaimplementowania serwera użyto języka C oraz API bsd-sockets. Klient oraz agent napisani zostali w języku python3. Aplikacja używa protokołu TCP/IP oraz wielowątkowości do zarządzania socketami , aby poprawnie obsługiwać współbieżną komunikacje.

2. Opis komunikacji pomiędzy serwerem i klientem

Server nasłuchuje nowych połączeń, czekając na połączenie klienta lub agenta. Maksymalna długość kolejki, ustawiona przez funkcję listen(), do jakiej liczba oczekujących połączeń może wzrosnąć wynosi 50. Na potrzeby mojej aplikacji jest to odpowiednia długość do poprawnego obsługiwania współbieżnej komunikacji. Gdy połączenie zostanie zaakceptowane, serwer oczekuje na dwubajtową informacje (‘c’ lub ‘a’) która definiuje czy połączenie pochodzi od klienta czy agenta. W zależności od tego, kolejne kroki serwera się różnią:

a. Klient – jeżeli otrzymana informacja to ‘c’, zaakceptowane połączenie pochodzi od klienta. Obsługa takiego połączeni zaczyna się od sprawdzenia czy tablica struktur przechowująca informacje każdego klienta nie została przepełniona. W razie potrzeby powiększana jest ona o kolejne 10 rekordów. Przy wykonywaniu tej operacji wypisywane na ekranie są informacje o wszystkich dotychczasowo połączonych klientach. Następnie tworzony jest nowy wątek do obsługi zaakceptowanego klienta. W wątku tym serwer oczekuje ponownie na dwubajtową informacje. Definiuje ona wykonywaną przez serwer funkcję. Istnieją trzy dostępne funkcjonalności:
    - Add (‘a’) – pozwala na dodanie nowych uprawnień klientowi. Serwer odbiera od klienta adres IP agenta. Następnie sprawdzane jest czy klient już nie ma przyznanych uprawnień do wskazanej maszyny oraz czy agent z podanego IP jest obecnie połączony. W przypadku błędu klientowi odsyłana jest odpowiednia informacja. W przeciwnym wypadku klient proszony jest o hasło dostępu do danego agenta. Jeżeli hasło jest poprawne przyznawane są uprawnienia, przy odpowiednim zarządzaniu pamięcią tzn. tablica przechowująca informacje o dostępach jest powiększana.

- Shutdown (‘d’) – pozwala na wyłączenie danej maszyny. Od klienta odbierany jest adres IP maszyny do wyłączenia. Sprawdzane jest czy agent z podanego IP jest obecnie połączony oraz czy klient posiada uprawniania do wyłączenia maszyny pod podanym adresem. W przypadku błędu odsyłana jest odpowiednia informacja. W przeciwnym wypadku do agenta (na odpowiedni socket) wysyłana jest wiadomość o wyłączeniu.

- Show(‘s’) – serwer przesyła do klienta informacje o wszystkich agentach. Wiadomość składa się z informacji o adresie agenta, informacji czy klient ma uprawnienia do wyłączenia danego agenta oraz informacji o stanie agenta tzn. 1 – jeżeli agent jest połączony, 0 – jeżeli agent jest rozłączony.

Gdy program klienta przestanie działać serwer wykryję to przez odebranie pustej wiadomości. W takiej sytuacji wątek używany do komunikacji z wyłączonym klientem przestaje działać.

b. Agent – jeżeli otrzymana informacja to ‘a’, zaakceptowane połączenie pochodzi od agenta. Obsługa takiego połączeni zaczyna się od sprawdzenia czy tablica struktur przechowująca informacje o agentach, nie została przepełniona. W razie potrzeby powiększana jest ona o kolejne 10 rekordów. Przy wykonywaniu tej operacji wypisywane na ekranie są informacje o wszystkich dotychczasowo połączonych agentach. Następnie przeszukiwane są połączenia od agentów. Jeżeli serwer wykryje że na próbującej się połączyć maszynie, działa obecnie już jeden agent, połączenie jest odrzucane, a do agenta wysyłana jest odpowiednia wiadomość. W przeciwnym przypadku tworzony jest nowy wątek, a do agenta wysyłana jest wiadomość o zaakceptowaniu połączenia. W wątku pobierane jest hasło dostępu, które pozwoli na weryfikacje klientów. Następnie połączenie z agentem utrzymywane jest w pętli nieskończonej. Wątek kończy swoje działanie gdy program agenta zostanie wyłączony. Wykrywane jest to przez otrzymanie pustej wiadomości od agenta.

3. Podsumowanie

- Implementacja przewiduje pełną walidacje danych wprowadzanych po stronie klienta oraz przesyłanych do serwera. Komunikacja z serwerem jest ograniczona i często opiera się na przesyłaniu dwubajtowych znaków, poprawnie rozszyfrowywanych po każdej ze stron komunikacji. Pozwala to przyspieszyć komunikacje oraz zmniejszyć przepływ informacji. W razie pojawienia się błędów lub braku dostępu, wyświetlane są czytelne dla użytkownika komunikaty. Komunikacja użytkownika z aplikacją odbywa się przez prostą konsolę obsługującą 5 poleceń z czego dwa z nich wymagają podania prawidłowej długości parametru. Program agenta można odpalić w tle, zalecane jest wtedy podawanie jako argumentu programu, hasła agenta.

- Trudność sprawiło pojawienie się w odebranej wiadomości atrybutów. Rozwiązaniem było czyszczenia tablicy przechowującej dane funkcją memset(). Problemem również okazało się zaimplementowania poprawnego zarządzania pamięcią oraz poprawnej walidacji danych. Zagnieżdżenie następujących po sobie operacji, oraz każdorazowe odbieranie i odsyłanie odpowiedniej widomości skutkowało dużą liczbą potrzebnych zmiennych do obsługi komunikacji. Ostatecznie starałem się ograniczyć ich liczbę do minimum często korzystając z dwubajtowych wiadomości oraz czyszcząc bufor.