05.04.2020

Dokumentacja projektu:
System komunikacji dla aplikacji kontroli dostępu do pomieszczeń

Realizują:
- **Zuzanna Żak**
- **Kacper Kipa - Lider**
- **Maciej Burda**
- **Marcin Kiliański**

Projekt wstępny:
1. Treść zadania:

System komunikacji pomiędzy serwerem, a zbiorem kart i sterowników otwierających drzwi w danej firmie.
Każdy pracownik posiada swoją własną kartę, która identyfikuje właściciela. Prawa dostępu do danych części budynku każdego pracownika przechowywane są w bazie serwera, a każda jego aktualizacja przekazywana jest do sterowników, które zarządzają otwieraniem odpowiednich drzwi. Sterowniki czujnika uwierzytelniają pracownika za pomocą karty oraz sprawdzają jego prawa dostępu, a także przesyłają te dane do serwera, który kontroluje oraz obsługuje takie rzeczy jak: liczba kart, liczba czujników czy prawa dostępu dla każdego użytkownika. Administrator ma możliwość dodawania kart i czujników, a także ustawiania praw dostępu dla konkretnych pracowników. System ponadto obsługuje takie funkcje jak:

- wykrywanie linii papilarnych użytkowników w celu podwójnego uwierzytelniania
- wykrywanie prób potencjalnego włamania:
  - próba użycia karty przy czujniku, do którego użytkownik nie ma praw dostępu

2. Nazwa własna projektowanego systemu:
StuSec - Student Security

3. Przyjęte założenia funkcjonalne i niefunkcjonalne:
Wymagania funkcjonalne:

- nawiązania i obsługa komunikacji dwustronnej serwer-sterowniki
- nawiązania i obsługa komunikacji pomiędzy serwerem, a bazą danych
- wysyłanie zadanego komunikatu w przypadku wykrycia niespójności w rejestrowanych danych lub awarii
- przesyłanie danych uzyskanych z czujników do sterownika

Wymagania niefunkcjonalne:

- racjonalny czas odpowiedzi systemu umożliwiający płynną pracę systemu - czujnik będzie weryfikował użytkowników w czasie nie dłuższym niż 3 sekundy

- system oraz czujniki będą odporne na standardowe ataki i próby podszycia się pod któreś z urządzeń
- system nie będzie udostępniał prywatnych, wewnętrznych danych nikomu kto nie posiada właściwych uprawnień
- oprogramowanie będzie mieć architekturę modułową
- możliwość dalszego rozwoju - system będzie można modyfikować skryptami w zależności od potrzeb

4. Podstawowe przypadki użycia:

Dodanie karty:

1. Administrator loguje się do systemu.
2. Administrator wprowadza do systemu identyfikator nowej karty.
3. System weryfikuje czy dana karta już istnieje, jeśli nie, zostaje ona dodana do bazy kart.
4. Podczas dodawania karty przez administratora i wprowadzeniu danych użytkownika (hasło, zeskanowanie linii papilarnych) system zapamiętuje wprowadzone dane i identyfikuje je z kartą. Następnie uzyskane dane zostają wysłane do serwera, a ten rozsyła je do każdego ze sterowników obsługujących czujniki w systemie.

Przepuszczenie użytkownika przez drzwi:

1. Pracownik podchodzi do czujnika i przykłada do niego kartę.
2. Czujnik prosi o ewentualne wpisanie kodu i/lub podanie odcisku palca.
3. Sterownik weryfikuje dane otrzymane przez czujnik oraz autoryzuje pracownika i jeśli proces przebiega poprawnie (poziom uprawnień, tożsamość pracownika) to otwiera drzwi, a dane o przejściu przesyła do serwera.

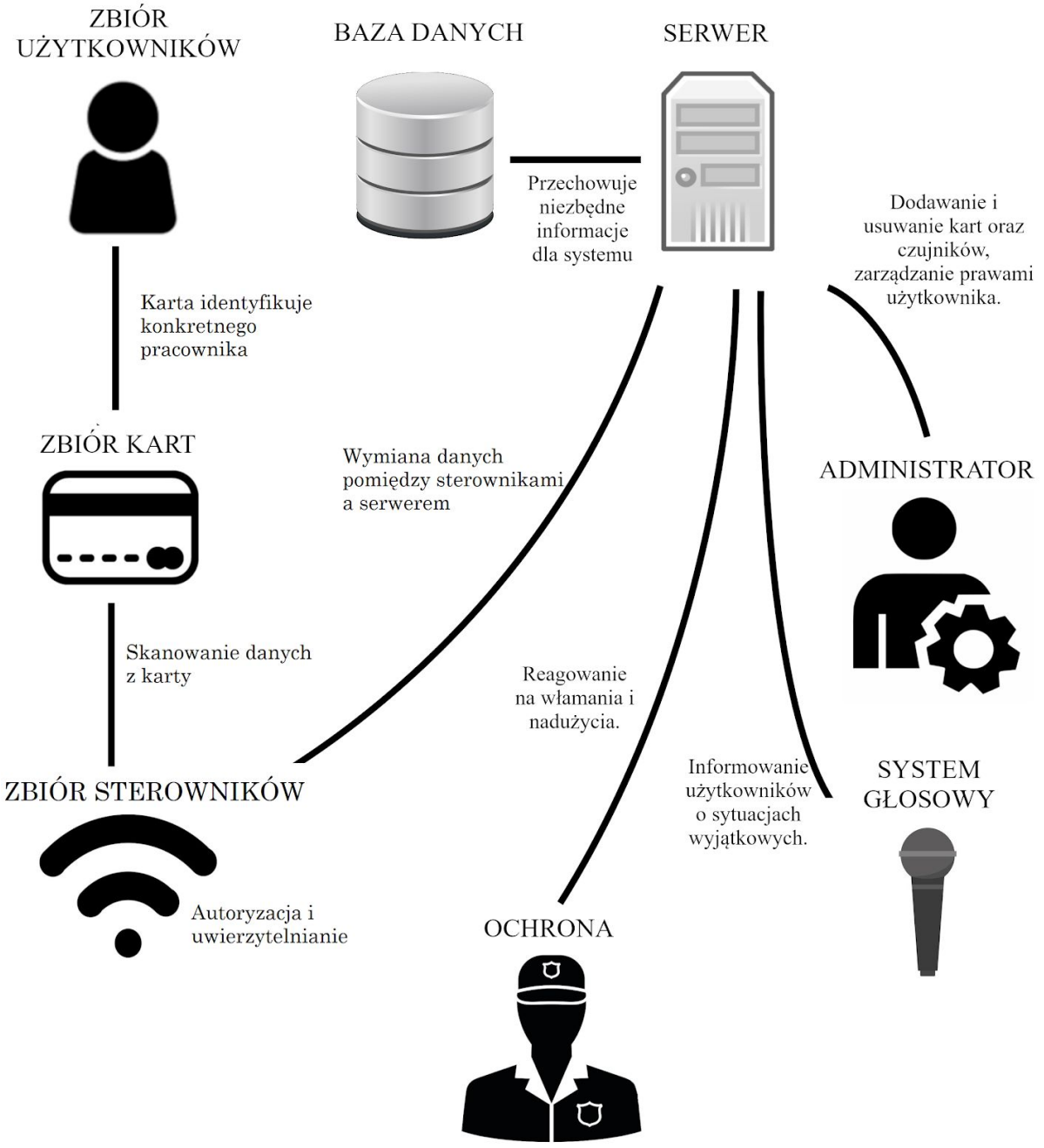
5. Wybrane środowisko sprzętowo-programowalne (systemy operacyjne, biblioteki programistyczne) i narzędziowe (testowanie, debugowanie):

Docelowym środowiskiem użytkowym programu będzie system Linux. Program będzie napisany w języku C++. System będzie korzystał przede wszystkim ze standardowych bibliotek takich jak:

- biblioteki służącę do komunikacji (np. sys/socket.h, netinet.h, boost.asio) - będą wspomagały komunikację pomiędzy poszczególnymi modułami systemu (serwer, czujniki, itd.)
- biblioteki bazodanowe (np. mysql) - będą wspomagały bezpieczny dostęp systemu do baz danych zawierających m.in. uprawnienia, hasła, czytniki, ID kart)
- biblioteki I/O (np. iostream) - obsługują podstawowe czynności administratora
- biblioteki szyfrujące (np. libgcrypt) - odpowiadają za bezpieczne szyfrowanie danych

Testowanie odbywało się będzie za pomocą testów “boost” z wykorzystaniem podstawowej biblioteki C++. Testowana będzie przede wszystkim integralność wszystkich modułów poprawność pod względem logicznym (np. dostęp odpowiednich pracowników do odpowiednich części budynku oraz wykrycia próby nieautoryzowanego dostępu). Do debugowania wykorzystywane będzie narzędzie “GDB”.

6. Architekturę rozwiązania, tj. ilustrację i opis struktury logicznej systemu (koncepcyjnych bloków funkcjonalnych):



7. Ewentualnie API modułów stanowiących bloki funkcjonalne:

Głównymi blokami funkcjonalnymi będzie blok sterowników, blok trzymający dane użytkowników (mała baza danych), a także blok będący głównym serwerem zarządzającym całością. Karty też będą stanowiły osobny blok jednak nie będzie on ściśle związany z podsystemem komunikacji.

Komunikacja pomiędzy blokiem sterowników a blokiem kart i serwerem:

Sterownik po wykryciu za pomocą czujnika zbliżenia do niego karty (proces symulowany) będzie odczytywał klucz (identyfikator) znajdujący się na karcie, a następnie analizował

zapisane w swojej pamięci informacje dotyczące niego. W zależności od praw dostępu do drzwi, program będzie dodatkowo prosił użytkownika o podanie kodu i/lub przyłożenie palca do czytnika, w celu weryfikacji odcisku palca. Wszystkie dane uzyskane powyżej przez czujnik będą w postaci tekstowej przekazywane do sterownika. Następnie sterownik będzie je szyfrował oraz uwierzytelniał, co pozwali mu zdecydować czy pracownik może czy nie może przejść. Dodatkowo zaszyfrowane dane, zostaną przesłane do modułu serwera odbierającego wszystkie informacje o przejściach przez drzwi pracowników i zbierającego te dane do analizy. Osobny moduł będzie łączył się z bazą danych i analizował otrzymane dane, wykrywał w nich niespójności oraz generował statystyki. W przypadku gdy administrator doda albo usunie kartę z systemu, informacja ta będzie przesyłana za pomocą serwera do wszystkich sterowników, które będą aktualizowały swoją pamięć.

Komunikacja pomiędzy blokiem kart a serwerem:

Jedynym innym przypadkiem komunikacji karty z serwerem będzie dodawanie i unieważnianie karty. Nie będzie to kontakt bezpośredni, ale na tyle ważny w systemie, że został podany jako osobny przypadek. Operacje dodawania i usuwania karty wykona administrator najpierw dodając kartę do serwera, a następnie przy pomocy sterownika, za pomocą którego, po podaniu kodu i odcisku palca przez pracownika oraz danych administratora (ponownie interpretowanych jako dane tekstowe) zostanie dokonana autoryzacja karty do określonych wcześniej przez administratora zasobów. Sterownik pierwsze użycie zinterpretuje jako priorytetowe i prześle komunikat do serwera, który roześle informacje o karcie i jej prawach do wszystkich sterowników i po otrzymaniu odpowiedzi z każdego karta będzie gotowa do użytku. Administrator nie będzie potrzebował pracownika ani jego karty w przypadku unieważnienia karty, a także zmiany praw dostępu pracownika.

Komunikacja serwera z bazą danych:

Serwer będzie jedynym komponentem struktury łączącym się z bazą danych. Będzie się to odbywało przy pomocy osobnego modułu współpracującego ze wszystkimi modułami serwera, a celem tego zachowania będzie weryfikacja danych w celu przekazania ich do modułu odpowiedzialnego za kontakt ze “światem zewnętrznym”.

8. Ewentualnie listy komunikatów z określeniem nadawców i odbiorców:

ADMIN -> SERWER:

Nazwy metod u admina i oznaczenia z jakimi są one identyfikowane przez serwer oraz informacje zwrotne, które otrzymuje administrator. Dodatkowo metody zawierają opis danych przesyłanych do serwera:

1. logIn:
   (login, hasło)
   - “A” - wszystko poprawnie
   - “l” - login nie istnieje (login albo hasło niepoprawne)
2. createAccount - “c”:
   (nazwa metody, login, hasło)
   - !“c” - błąd metody
   - “A” - wszystko poprawnie
   - “l” - login już istnieje
   - “p” - login lub hasło nie spełniają wymogów
3. deleteAccount - “C”:
   - !“C” - błąd metody
   - “A” - wszystko poprawnie

- “l” - login nie istnieje więc nie może być usunięty

4. addCard - “a”:
   (nazwa metody, ID karty, kod użytkownika, skan palca)
   - !“a” - błąd metody
   - “A” - wszystko poprawnie
   - “i” - kart o podanym ID już jest w systemie
   - “c” - kod nie spełnia wymogów
   - “f” - skan palca nie spełnia wymogów lub jest w systemie
5. deleteCard - “A”:
   - !“A” - błąd metody
   - “A” - wszystko poprawnie
   - “i” - ID karty nie istnieje więc nie może być usunięte
6. setAccessRights - “s”:
   - !”s” - błąd metody
   - “A” - wszystko poprawnie
   - “i” - ID karty nie istnieje więc nie można ustawić jej priorytetu
   - “l” - ustawiany poziom dostępu nie spełnia wymogów
7. addDriver - “d”:
   - !”d” - błąd metody
   - “A” - wszystko poprawnie
   - “i” - ID sterownika już istnieje więc nie można go dodać
   - “p” - ID sterownika lub hasło są niepoprawne
   - “l” - poziom dostępu jest poza zakresem
8. deleteDriver - “D”:
   - !“D” - błąd metody
   - “A” - wszystko poprawnie
   - “i” - ID sterownika nie istnieje więc nie może być usunięte
9. logOut

SERWER -> ADMIN:

Podczas logowania:

- "login" - błędny login (oczekuje podania jeszcze raz loginu i hasła)
- “passwd" - błędne hasło (oczekuje podania jeszcze raz loginu i hasła)
- "Admin Authenticated" - dane poprawne (admin zalogowany, uprawniony)

Wybór metody:

- “Creating Account” (“c”) - przejście do tworzenia konta (admina)
- “Adding Card” (“a”) - przejście do tworzenia karty (pracownika)
- “error” - błędny kod metody (oczekuje podania jeszcze raz kodu metody)

Podczas tworzenia konta:

- "login" - błędny login (powrót do wyboru metody)
- “passwd" - błędne hasło (powrót do wyboru metody)
- "Admin Authenticated" - dane poprawne (nowy admin stworzony / powrót do wyboru metody)

Podczas dodawania karty:

- “idCard” - błędne ID karty; błąd karty (powrót do wyboru metody)
- “code” - błędny kod pracownika (powrót do wyboru metody)
- “finger” - błąd odcisku palca (powrót do wyboru metody)

- “Adding card successfully” - dane poprawne; karta dodana (powrót do wyboru metody)

Podczas usuwania karty:
- "idCard" - niepoprawne ID karty
- “Card deleted" - wszystko poprawnie

Podczas zmieniania praw dostępu karty:
- “idCard” - błędne ID karty lub hasło
- "levelOfPriority" - priorytet poza zakresem
- “New access rights set” - wszystko poprawnie

Podczas dodawania sterownika:
- "idDriver" - ID sterownika już istnieje w bazie
- "passwd" - niepoprawne hasło lub ID sterownika
- "levelOfPriority" - błędny zakres praw dostęp
- "New driver added!" - wszystko poprawnie

Podczas usuwania sterownika:
- "idDriver" - ID sterownika nie istnieje w bazie danych
- "Driver deleted" - wszystko poprawnie

SERWER -> STEROWNIK:

Podczas logowania:
- "login" - błędny login (oczekuje podania jeszcze raz loginu i hasła)
- “passwd" - błędne hasło (oczekuje podania jeszcze raz loginu i hasła)
- "Authenticated" - dane poprawne (sterownik zalogowany, nasłuchiwany)

Broadcast:
- serwer przesyła kolejno dane na temat kolejnych kart użytkowników

STEROWNIK ->SERWER:

Podczas pracy:
- “b” - sterownik gotowy do broadcastu (otrzyma zaktualizowane dane kart)
- “s” - sterownik zgłasza przejście przez drzwi (kod drzwi, kod karty, wynik próby)

9. Sposób testowania:

Do systemu zostanie na początku dodana pewna pula kart (liczba pracowników). Następnie zostaną przygotowane pliki testowe, które będą zawierały imitacje działań systemu takie jak: kolejne komunikaty wysyłane przez czujniki, dodanie kart, unieważnienie kart itd. Na podstawie tych testów będą sprawdzane reakcje systemu, w postaci wywołań komunikatów z serwera na ekran, które będą również zapisywane w pliku z rezultatami.

10. Sposób demonstracji rezultatów, tj. scenariusze testów akceptacyjnych do zaprezentowania przy odbiorze projektu:

Scenariusz 1. Poprawne działanie wszystkich pracowników:
1. Serwer nie wykrywa żadnego zagrożenia i stale informuje o użyciu danych drzwi przez pracowników.
2. Na konsoli wyświetlają się komunikaty informujące o identyfikatorze czujnika i identyfikatorze karty.

11. Podział prac w zespole:

Marcin Kiliański:
- budowa i obsługa baz danych w sterownikach jak i w serwerze
- tworzenie testów i wstępnych danych do baz

Maciej Burda:
- implementacja serwera od strony obsługi admina
- implementacja sterowników

Zuzanna Żak:
- stworzenie funkcji haszującej
- stworzenie grafu stanów i przypadków użycia

Kacper Kipa:
- implementacja obsługi admina
- implementacja serwera od strony obsługi sterowników

12. Harmonogram prac:

Wstępne założenia:
30.04 - Wstępna implementacja serwera, administratora zarządzającego kartami (dodawanie, usuwanie) oraz bazy danych przechowującej dane.
26.05 - Implementacja sterowników oraz modułu odpowiedzialnego za weryfikację użycia karty.

13. Adres projektu na serwerze kontroli wersji:

https://github.com/Kicper/TIN_Cpp

Ilustrację i opis struktury implementacyjnej systemu, a także definicje komunikatów:

W pliku “Opis_przypadków_użycia.pdf” zamieszczonym w powyższym repozytorium znajdują się opisy komunikatów i przypadków użycia realizowanych przez nasz system.

Wnioski z wykonanego testowania:

Po wykonanych testach możemy stwierdzić, że baza działa poprawnie. Może obsługiwać jednocześnie wiele sterowników, jednocześnie mając otwarte osobne gniazdo dla administratorów. Przed jakąkolwiek codzienną komunikacją (zgłaszanie przejść, edycja kart) pomiędzy dowolnym modułem a serwerem wymagana jest autoryzacja co zapobiega niechcianym ingerencją w serwer z zewnątrz. Dane są bezpiecznie przechowywane w bazie danych, która pozwala na stały dostęp do nich autoryzowanym osobom, a system spełnia wszystkie założenia czasowe wykonując narzucone mu zadania w odpowiednio krótkim kwancie czasu.

Analiza podatności bezpieczeństwa:

Serwer jest zabezpieczony przed podszywaniem się pod jakikolwiek sterownik albo admina przez osoby postronne. Jest odporny zarówno na przepełnienie bufora jak i SQL injection

(przede wszystkim podczas logowania, czyli próby uzyskania dostępu przez osobę postronną). Jeśli otrzyma komunikaty, na które teoretycznie nie jest przygotowany (inne pod względem ilości np. niż ze strony sterownika) to w najgorszym przypadku zawiesi dane połączenie. Natomiast nie będzie miało to żadnego wpływu na pozostałe połączenia ze sterownikami oraz administratorami.

Instrukcja instalacji i dezinstalacji stworzonego systemu:

Aby system mógł poprawnie działać należy utworzyć dwie bazy danych wraz z odpowiednimi tabelami i ewentualnie dodać odpowiednie rekordy. Wszystkie informacje dotyczące tego jak to zrobić można znaleźć w pliku “InstukcjaDoBazTIN.pdf”, który znajduje się w repozytorium: https://github.com/Kicper/TIN_Cpp. Należy też w kodzie programu zmienić nazwę użytkownika i hasło aby ustawić połączenie pomiędzy bazą tworzoną w programie, a bazą stworzoną przez użytkownika.

Podsumowanie:

Podczas realizacji projektu mieliśmy okazję nauczyć się wielu przydatnych rzeczy przy obsłudze gniazd BSD, a także umiejętności związanych z tworzeniem sieci i obsługą ich przez programy. Przy tworzeniu tego rodzaju aplikacji nauczyliśmy się, że warto jest mieć wyraźnie odseparowane części projektu, które pozwalają na niezależną pracę wszystkich członków zespołu. Szczególnie ważne były testy integracyjne, które były przeprowadzane dość często (1-2 razy w tygodniu), aby sprawdzić czy moduły nadal odpowiednio ze sobą współpracują. W celu poprawy pracy w projekcie warto było na początku stworzyć pełną listę sytuacji do zaimplementowania (wraz z komunikatami) oraz konkretną budowę każdego modułu (przede wszystkim tabel w bazach danych). Projekt był bardzo czasochłonny więc spotkania kontrolne pomogły rozplanować działania w czasie oraz motywowały nas do systematycznej pracy.

Projekt ostatecznie składa się z czterech modułów: serwera, sterownika, funkcji haszującej oraz administratora. Moduł administratora jak i sterownika są zbiorami kilku plików średniej długości. Największy rozmiar ma moduł serwera, a co za tym idzie, najwięcej czasu zostało poświęconego na stworze go oraz odpowiednią implementację w nim bazy danych jak i wszystkich komunikatów. Finalnie czas przeznaczony na projekt wynosił około 280 godzin.