

Projekt TIN

Dokumentacja końcowa

Autorzy: Kunikowski Bartłomiej, Saramonowicz Przemysław, Ostapowicz Michał, Rubin Julian

1. Cel zadania

Celem zadania była implementacja systemu zarządzania zdalnym dostępem do urządzenia. System umożliwia:

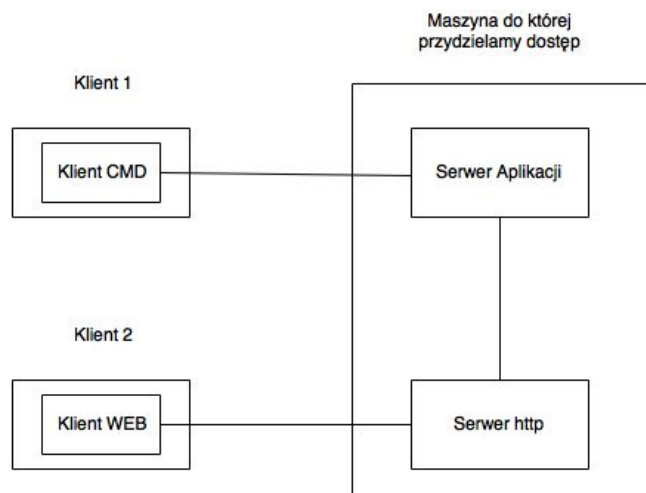
- rezerwację terminu dostępu do urządzenia
- anulowanie rezerwacji
- odblokowanie adresu IP po rozpoczęciu czasu trwania rezerwacji

Dostęp do systemu jest możliwy przez klienta tekstowego i aplikację internetową.

2. Architektura

2.1. Serwer HTTP

Serwer apache2 z warstwą logiczną i wizualną zaimplementowaną przy pomocy frameworka Django jest odpowiedzialna za pośredniczenie w komunikacji pomiędzy serwerem aplikacji i przeglądarką internetową. Komunikacja między serwerem a użytkownikiem przebiega z użyciem protokołu HTTPS. Serwer HTTP po uwierzytelnieniu będzie komunikował się z serwerem aplikacji przy użyciu gniazd i protokołu opisanego w punkcie 3.



2.2. Serwer aplikacji

Serwer aplikacji wykorzystuje do komunikacji gniazda TCP. Jego zadaniem, jest przetwarzanie danych otrzymywanych z serwera http od klienta, synchronizacja danych i zarządzanie firewallem na urządzeniu poprzez skrypty z poleceniami iptables. Wygaśnięcie sesji użytkownika, jest realizowane poprzez zablokowanie jego IP w iptables.

Modyfikacja wpisów w iptables odbywa się poprzez wywołanie odpowiedniego skryptu poleceniem system() :

"iptables -A input -p tcp --dport 22 -s <IP> -j ACCEPT" - dla dodawania

"iptables -D input -p tcp --dport 22 -s <IP> -j ACCEPT" - dla usuwania

Dla każdego klienta, który łączy się z serwerem tworzony jest nowy wątek.

W momencie kiedy klient chce otworzyć dostęp do SSH, wątek, który zarządza czasem dostępu do SSH, w momencie wygaśnięcia czasu rezerwacji usuwa regułę z iptables.

Serwer aplikacji będzie synchronizował dostęp do pliku z logiem i z rezerwacjami.

Zdecydowaliśmy się na skorzystanie z protokołu TCP. Zakładamy, że liczba użytkowników końcowych nie będzie duża. Z tego powodu możemy pozwolić sobie na utrzymywanie wielu połączeń jednocześnie. (patrz pkt. 5)

Nie zdecydowaliśmy się na protokół UDP z następujących przyczyn:

1. Serwer musiałby uwierzytelniać klienta przy każdej wiadomości.
2. Musielibyśmy wziąć pod uwagę sytuację, w którym pakiet UDP zostałby zgubiony. W przypadku takiego zdarzenia musielibyśmy przyjąć jakiś czas jaki dajemy serwerowi na odpowiedź, a w razie jej braku ponowić wiadomość.

2.3. Biblioteki pomocnicze

SHA3, Json for Modern C++.

2.4. Klient (linia poleceń)

Klient konsolowy łączy się bezpośrednio z aplikacją serwerową. Dostarcza tych samych funkcjonalności, co klient http. Uwierzytelnia użytkownika przy pomocy protokołu challenge-response.

3. Komunikaty

Format komunikatów, przesyłanych pomiędzy klientami i serwerem to JSON. Kontrola poprawności otrzymanych wiadomości opiera się na zliczaniu klamr (za pomocą stosu). Poniżej przedstawiono obsługiwane komunikaty.

Oznaczenia

- **Datetime:** dzień.miesiąc.rok godzina:minuta:sekunda
- **dzień:** 1 - 31 | 01 - 31
- **miesiąc:** 1 - 12 | 01 - 12
- **rok:** pełny, np. 1994, 2000 etc.
- **czas:** np. 18:00:15
- **Duration:** dodatnia liczba całkowita
- **Login:** ciąg znaków alfanumerycznych (zliczanie klamr uniemożliwia korzystanie ze znaków ' { ' i ' } ')

Przesyłający	Komunikat
1. Uwierzytelnienie	
Klient	<pre>{ "msg": "handshake", "login": Login }</pre>
Serwer	<pre>{ "msg": "challenge", "challenge": "... }</pre>
Klient	<pre>{ "msg": "response", "password": <hash of password and challenge> }</pre>
Serwer	<pre>{ "msg": "authenticated", "value": true/false }</pre>
2. Rezerwacja terminu	
Klient	<pre>{ "msg": "reservation", "start": Datetime, "duration": Duration }</pre>
Serwer	<pre>{ "msg": "reserved", "value": true/false, "overlap": { "start": Datetime, "duration": Duration } }</pre>
3. Anulowanie rezerwacji	
Klient	<pre>{ "msg": "cancel", "start": Datetime }</pre>
Serwer	<pre>{ "msg": "canceled", "value": true/false }</pre>

4. Odblokowanie IP	
Klient	<pre>{ "msg": "unlock", "ip": ipaddress }</pre>
Serwer	<pre>{ "msg": "unlocked", "value": true/false }</pre>
5. Moje rezerwacje	
Klient	<pre>{ "msg": "getMyReservations" }</pre>
Serwer	<pre>{ "msg": "calendar", "reservations": [{ "start": Datetime, "duration": Duration, "username": Login }, other reservations ...] }</pre>
6. Wszystkie rezerwacje	
Klient	<pre>{ "msg": "getCalendar" }</pre>
Serwer	<pre>{ "msg": "calendar", "reservations": [{ "start": Datetime, "duration": Duration, "username": Login }, other reservations ...] }</pre>

4. Przechowywanie danych użytkowników

Loginy i hasła użytkowników będą przechowywane w pliku. Hasło będzie przetwarzane w formie hasha, żeby zabezpieczyć plik w razie jego wykradzenia. Użyjemy funkcji skrótu SHA-3 256b. Formatem pliku jest JSON.

Przykładowy fragment pliku z danymi użytkowników:

```
{
  "users": [
    {
      "login": "mostapow",
      "password_hash":
      "b65f0f7665bc62f898d5e8594ab2642740eeb38e769a21ba68d36387a3b4a0c8"
    },
    {
      "login": "psaramon",
      "password_hash":
      "50726529251adcf6a2cf97f3104893789c864cc7ad9048a148ead3fd7a8d1a26"
    }
  ]
}
```

5. Zrealizowane funkcjonalności

- możliwość rezerwacji czasu zdalnego dostępu do maszyny
- odblokowanie adresu IP użytkownika, który wcześniej zarezerwował dostęp
- blokowanie IP użytkownika po upływie zarezerwowanego czasu
- usuwanie rezerwacji i anulowanie odblokowania IP

6. Wielowątkowość i synchronizacja

Kiedy serwer aplikacji akceptuje nowe połączenie z klientem, tworzy nowy wątek, który odpowiada za bieżącą sesję użytkownika. Skorzystamy z wątków pthread.

Plik z danymi użytkowników jest wykorzystywany w trybie tylko do odczytu i nie wymaga synchronizacji.

Plik z logami i dostęp do kalendarza jest synchronizowany. Używamy do tego mechanizmu wzajemnego wykluczania (klasa pthread_mutex_t).

Po wpisaniu użytkownika do iptables (udostępnienia usługi SSH) zostaje uruchomiony nowy wątek, który będzie zawieszany do czasu, w którym będzie wymagane zablokowanie usługi SSH dla użytkownika. Zrealizowane zostanie to za pomocą funkcji sleep z biblioteki standardowej.

Wczytanie kalendarza na stronie internetowej będzie odbywać się za pośrednictwem nowego wątku : workera.

7. Logowanie serwera aplikacji

Logi będą gromadzone na podstawie działania serwera aplikacji w formacie CSV gdzie każda z "sekcji" będzie podzielona ";" a każdy nowy log będzie zaczynał się od nowej linii. Będzie on logował wiadomości przesyłane do niego z serwera http, od klienta konsolowego, a także generowane przez sam serwer.

Format logów:

Data, godzina; Adres; Użytkownik; Typ; Opis

```
00:00:00/01/01/2016; 192.168.0.1; user1; New Year!; Happy New Year!  
23:59:59/31/12/2015; 192.168.0.2; user2; Last Year!; Happy Last Year!  
23:59:58/31/12/2015; 192.168.0.3; user2; Hello!; World!
```

Opis pól:

Data, godzina	Format: hh:mm:ss/dd/MM/yyyy
Adres	Adres IP połączenia którego ten log dotyczy
Użytkownik	Nazwa użytkownika którego ten log dotyczy
Typ	Rodzaj zapisywanego zdarzenia p. tabela poniżej
Opis	Dodatkowe informacje o danym logu zależne od typu logu.

Typ	Opis	Cel
INITIALIZE		inicjalizacja
HANDSHAKE	login	próba logowania
AUTHENTICATED	login	użytkownik uwierzytelniony
NOT_AUTHENTICATED	login	błąd uwierzytelnienia
RESERVATION	date, duration	próba rezerwacji
RESERVED		zarezerwowano
NOT_RESERVED	date, duration	nie zarezerwowano
UNLOCK		odblokuj IP
UNLOCKED		odblokowane
NOT_UNLOCKED		nie odblokowane
GET_CALENDAR		żądanie wysłania kalendarza

CALENDAR		kalendarz wysłany
CANCEL	date	anulowanie rezerwacji
CANCELED	date	rezerwacja anulowana
IPTABLES_TIMEOUT	pole adres wypełnione IP z iptables; pole użytkownik wypełnione "SERVER"	zakończono sesję SSH : skończył się czas
BAD_CALL	pełna treść wysłanej wiadomości	nieprzewidziana w protokole komenda
SESSION_END		koniec sesji użytkownika

Administrator serwera aplikacji powinien sam dbać o archiwizację plików logu i zarezerwowanych dat.

7. Parametry działania programu

Każdy moduł będzie posiadał własny plik konfiguracyjny:

- aplikacja kliencka: adres serwera, port serwera

```
{
  "user_configuration": {
    "server_ip": "192.168.12.31",
    "port": "8000"
  }
}
```

- aplikacja serwera: port, ścieżka do pliku z hasłami użytkowników, ścieżka do pliku z datami rezerwacji, ścieżka do pliku dziennika

```
{
  "configuration": {
    "port": "8000",
    "user_file_path": "/Server/users.txt",
    "calendar_file_path": "/Server/calendar.json",
    "log_file_path": "/Log/serverLog.csv"
  }
}
```

Zatwierdzone rezerwacje są przechowywane w pliku JSON o następującym formacie:

```
{
  "reservations": [
    {
      "date": "12.03.2016",
      "start_hour": "12:45",
      "interval_in_hours": "2",
      "username": "jrubin"
    },
    {
      "date": "13.03.2016",
      "start_hour": "14:00",
      "interval_in_hours": "3",
      "username": "psaramon"
    }
  ]
}
```

8. Obsługa sytuacji wyjątkowych

8.1. Serwer aplikacji jest niedostępny przy próbie nawiązania połączenia przez klienta

Wyświetlenie w kliencie komunikatu o braku możliwości połączenia z serwerem

8.2. Serwer aplikacji przestaje działać w trakcie sesji

Wyświetlenie w kliencie komunikatu o zerwaniu połączenia i usunięcie sesji.

8.3. Nieprawidłowe dane wejściowe w kliencie (nieprawidłowy format danych, np. zamiast godziny 10:aa), próba rezerwacji zajętego terminu, próba odblokowania ip bez wcześniejszej rezerwacji.

Odrzucenie danych i wyświetlenie komunikatu oraz zapisanie informacji w logu. Klient konsolowy jedynie wyświetla komunikat o błędzie, a niektórych sytuacjach prosi o ponowne wprowadzenie danych.

8.4. Przywracanie domyślnych ustawień iptables, kiedy maszyna, na której jest uruchomiony serwer aplikacji, wyłączy się (brak prądu, naciśnięcie przycisku ponownego uruchomienia)

Zmiany w iptables są nietrwałe, tzn. dopóki nie korzystamy z polecenia iptables-save, zmiany pozostają w mocy tylko do czasu ponownego uruchomienia maszyny. Oznacza to, że zawsze po ponownym uruchomieniu maszyny, iptables będzie miało konfigurację domyślną (otwarty port SSH i serwera aplikacji).

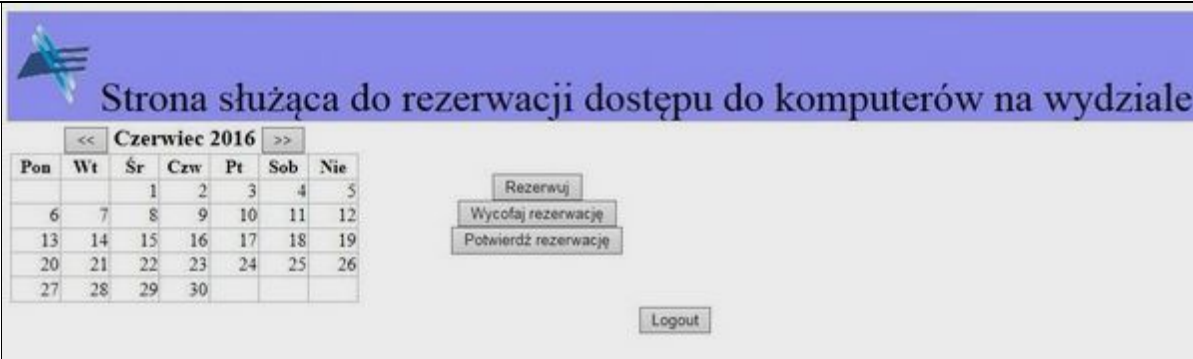
8.5. Przywracanie sesji klienta w przypadku utracenia połączenia z serwerem (problemy po stronie użytkownika, serwer dalej działa)

Sesja nie jest w żaden sposób przywracana. Jeżeli użytkownik nie otrzyma od serwera potwierdzenia rezerwacji / odblokowania, nie może zakładać, że jego działania przyniosły pożądany rezultat po stronie serwera.

8.6. Nieprawidłowe dane podczas uwierzytelnienia

W przypadku niepowodzenia uwierzytelnienia, wyświetlany jest komunikat o błędzie uwierzytelnienia (zawsze taki sam, żeby utrudnić wykradanie loginów)

9. Aplikacja przeglądarkowa



Strona służąca do rezerwacji dostępu do komputerów na wydziale

<< Czerwiec 2016 >>

Pon	Wt	Śr	Czw	Pt	Sob	Nie
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

Rezerwuj

Wycofaj rezerwację

Potwierdź rezerwację

Logout

Okno służące do rezerwowania terminów

Aby zarezerwować dany termin, klikamy na wybranej dacie i przytrzymując przycisk myszy zaznaczamy na liście godziny, które nas interesują. Następnie wciskamy przycisk Rezerwuj. Gdy chcemy wycofać rezerwację postępujemy w analogiczny sposób. Terminy zarezerwowane przez nas oznaczone są kolorem zielonym, natomiast zarezerwowane przez innych użytkowników - czerwonym.