Uniwersytet w Białymstoku

Wydział Matematyczno-Fizyczny

Instytut Matematyki

Robert Tokarski

Dostosowanie serwera Teapop do obsługi wielu domen

Praca dyplomowa napisana
pod kierunkiem
dr. Mariusza Żynela

Białystok 2004

Składam serdeczne podziękowania
dr. Mariuszowi Żynelowi za pomoc
podczas przygotowania niniejszej pracy

Robert Tokarski

Spis treści

Wstęp

W chwili obecnej można zaryzykować stwierdzenie, że nie ma osoby, która nie słyszałaby o Internecie. Trudno powiedzieć ile dokładnie osób korzysta z Internetu, ale dla wielu ludzi na pewno jest on niezbywalnym narzędziem pracy.

Określenie Internet kojarzone jest ze stronami WWW i pocztą elektroniczną. Z założenia Internet, początkowo nazywany ARPANET, miał służyć do wymiany wiadomości, czyli tego co dzisiaj nazywamy pocztą e-mail. W chwili obecnej w Internecie mamy mnóstwo różnych usług takich jak: P2P (Kazaa, Gnutella, Direct Connect), IRC, Gadu-Gadu.

Z technicznego punktu widzenia poczta elektroniczna to dwa ze sobą związane protokoły: jeden z nich służy do wysyłania poczty (nazywamy go SMTP), natomiast drugi służy do odbierania wiadomości z serwera (może to być POP3 albo IMAP). Moja praca poświęcona jest jednemu z programów realizujących usługę POP3, a mianowicie serwerowi Teapop.

Praktyczna część mojej pracy polegała na dokonaniu zmian w programie Teapop, tak aby w jednej tabeli bazy danych MySQL można było przechowywać opis kont pocztowych z wielu różnych domen oraz na napisaniu skryptów do zarządzania kontami wirtualnymi POP3. Po przeczytaniu pracy może powstać wrażenie, że część pierwsza była niemal trywialna, gdyż formalne modyfikacje są nieliczne. Aby jednak wykonać te modyfikacje należało wcześniej rozpoznać sposób działania serwera Teapop na poziomie implementacyjnym, inaczej mówiąc należało przeczytać kod źródłowy tego oprogramowania.

Serwer Teapop z naniesionymi przeze mnie zmianami oraz skrypty administracyjne są wykorzystywany w praktyce na kilku komercyjnych serwerach pocztowych w Białymstoku, w firmie Amikom, BCMB, Solution oraz na serwerze Instytutu Matematyki.

Rozdział 1

Protokół POP3

Na którym komputerze powinna się znajdować skrzynka pocztowa? Użytkownicy mający możliwość wyboru wolą mieć swoją skrzynkę na tym komputerze, którego najczęściej używają. Niestety skrzynka pocztowa nie może rezydować na dowolnym komputerze, a jedynie na takim, który bezustannie wykonuje program serwera pocztowego (zwany serwerem SMTP, np. *Sendmail* [8] albo *Qmail* [7]). Rzecz jasna, że taki komputer musi też być bezustannie przyłączony do Internetu, bądź pewnej sieci w obrębie, której poczta ma być wymieniana. Wymyślono pewne rozwiązanie, które umożliwia pobieranie poczty ze skrzynki pocztowej umieszczonej na takim komputerze do lokalnej stacji roboczej. Mianowicie jest to protokół POP (*Post Office Protocol*). Wymaga on by na maszynie ze skrzynkami pocztowymi działał dodatkowo serwer POP.

W powszechnym użyciu są dwie wersje tego protokołu: POP2 i POP3. Wersja POP2 została zdefiniowana w RFC 937 [5], a POP3 w RFC 1725 [6] i RFC 1939 [6]. POP2 używa portu 109, a POP3 portu 110. Wersje są niezgodne ze sobą, używają innych komend, choć wykonują te same funkcje. Najbardziej rozpowszechniony jest POP3.

Krótko mówiąc POP3 jest odpowiedzialny za odbieranie poczty elektronicznej z serwera, tak by ta trafiła na lokalny komputer. W większości przypadków operację tę realizują programy pocztowe, np. *Netscape/Mozilla, Outlook, Kmail*, albo specjalizowane aplikacje do odbierania poczty, takie jak *Fetchmail* czy *Fetchpop*. Rola użytkownika ogranicza się jedynie do podania nazwy serwera POP3, z którego program ma skorzystać oraz do dokonania autoryzacji.

Lepsze poznanie protokołu POP pozwoli na zrozumienie działania serwera Teapop [10] i pomoże w analizie jego kodu źródłowego. Poczta trafia na konta użytkowników na odległych serwerach za pomocą protokołu SMTP - protokołu wykorzystywanego przy przesyłaniu poczty między maszynami. Aby móc ją pobrać na lokalny komputer, wykorzystujemy POP3 - wiadomości są transmitowane wtedy, gdy tego zażądamy. Rola protokołu POP3 ogranicza się do pobierania wiadomości i kasowania ich z serwera. Bardziej zaawansowane i kompleksowe mechanizmy operujące na skrzynkach pocztowych oferuje protokół IMAP.

Głównym zadaniem IMAP (*Internet Message Access Protocol*) jest umożliwienie stacji roboczej (np. pecet w domu lub w sieci lokalnej) dostępu do listów elektronicznych znajdujących się w skrzynce pocztowej (*mail box*) na serwerze pocztowym. Protokół POP zajmuje się tym samym problemem ale jego możliwości są znacznie mniejsze. Podobnie jak POP, IMAP definiuje środki dostępu do listów w skrzynce, a nie środki wysyłania czy transferu listów pomiędzy serwerami pocztowymi. Pocztowy protokół dostępu IMAP podobnie jak POP, odbiera wiadomości ze skrzynki pocztowej na serwerze i przekazuje je do stacji roboczej. Ponadto IMAP może modyfikować atrybuty listów; w ogólności zarządza skrzynkami. IMAP oferuje wiele więcej możliwości w stosunku do prostego schematu "skopiuj i usuń" dostępnego w protokole POP.

1.1 Ogólna idea działania protokołu POP3

Usługa POP3 w systemie wykorzystuje port TCP o numerze 110. W momencie kiedy klient POP chce skorzystać z usługi, zestawiane jest połączenie klient-serwer. Następnie serwer wysyła komunikat powitalny i rozpoczyna się sesja wymiany poleceń oraz odpowiedzi. Trwa ona dopóki połączenie nie zostanie zamknięte lub przerwane. Polecenia w POP3 są to wyrażenia o nieistotnej wielkości liter. Mogą one przyjmować jeden lub więcej argumentów. Zarówno nazwy poleceń jak i argumenty muszą składać się ze znaków ASCII, a rozdzielane są znakiem spacji. Nazwy poleceń mają zazwyczaj długość trzech lub czterech znaków, natomiast każdy z argumentów nie może przekroczyć długości 40-tu znaków.

Odpowiedzi w protokole POP3 składają się ze wskaźnika stanu i opcjonalnych dodatkowych informacji. Wszystkie odpowiedzi nie mogą przekroczyć 512 znaków. POP3 dysponuje dwoma wskaźnikami stanu odpowiedzi: pozytywnym `+OK` i negatywnym `-ERR`. Oba pisane są wielkimi literami. Odpowiedzi na niektóre polecenia są wieloliniowe. W takim przypadku wskaźnik statusu występuje w pierwszej linii odpowiedzi, jeszcze w tej samej linii jak i w następnych liniach mogą występować dodatkowe informacje. Całość zakończona jest kropką.

Serwer POP3 musi odpowiadać również na polecenia nierozpoznawalne lub niepoprawne składniowo. Odpowiada wówczas negatywnym wskaźnikiem stanu. Tutaj klient zazwyczaj nie jest w stanie stwierdzić różnicy między odpowiedzią serwera oznaczającą, że polecenie jest niezaimplementowane, a odpowiedzią, że serwer nie jest w stanie wykonać polecenia (np. z powodu braku zasobów).

Serwer POP3 posiada mechanizm automatycznego wylogowywania. Serwer po nieotrzymaniu żadnego polecenia w ciągu co najmniej dziesięciu minut zamyka połączenie TCP. Nie usuwa przy tym żadnych wiadomości ze skrzynki pocztowej oraz nie wysyła żadnej odpowiedzi do klienta.

1.2 Stany pracy protokołu POP3

Opisując protokół POP3, należy wspomnieć o trzech rozróżnialnych stanach pracy z serwerem. Podczas logowania do systemu znajdujemy się w stanie autoryzacji (authorization state). Po podaniu nazwy użytkownika i hasła oraz po zweryfikowaniu ich przez serwer przechodzimy do stanu transakcji (transaction state). Za zmianę statusu skrzynki pocztowej odpowiada ostatni ze stanów - stan aktualizacji (update state). Przy rozłączaniu się z usługą zaznaczone do usunięcia wiadomości są fizycznie kasowane ze skrzynki pocztowej.

1.2.1 Stan autoryzacji

Zanim zaczniemy pracować na serwerze musimy się z nim połączyć oraz uzyskać dostęp. Ta faza zwana jest stanem autoryzacji. Polecenie nawiązujące połączenie TCP może wyglądać następująco:

```
telnet 192.168.0.1 110
```

gdzie 192.168.0.1 jest adresem serwera POP (oczywiście możemy podawać nazwy symboliczne), a 110 - numerem portu.
Po nawiązaniu połączenia serwer odpowiada jednoliniowym komunikatem powitalnym o pozytywnym wskaźniku stanu, na przykład:

```
+OK POP3 server ready
```

Od tego momentu sesja POP3 znajduje się w stanie autoryzacji. Klient musi dokonać swojej autoryzacji przed serwerem. Istnieją dwa mechanizmy używane do tego celu: polecenia USER i PASS, albo drugi - polecenie APOP. Należy pamiętać, że na zalogowanie się do systemu mamy określony czas. Po jego przekroczeniu serwer zakończy połączenie. Ta sama zasady dotyczy aktywności użytkownika. W razie niewykonywania żadnej czynności serwer zakończy sesję automatycznie.

Istotną wadą protokołu POP3 jest przekazywanie hasła otwartym tekstem. O ile możliwe jest podejrzenie haseł przez sieć, o tyle przy korzystaniu z tandemu *telnet+pop3* powinniśmy mieć na uwadze fakt, że wpisywane po komendzie PASS hasło jest widoczne na ekranie. Wówczas trzeba uważać na to, kto patrzy na monitor zza naszych pleców. Po zakończonym powodzeniem procesie autoryzacji, klient powinien mieć dostęp do odpowiedniej skrzynki pocztowej. Serwer POP3 rozpoczyna pracę ze skrzynką w trybie wyłączności. Tryb ten zapobiega przed modyfikacją lub usunięciem wiadomości przed przejściem w tryb aktualizacji. Jeśli przejście w tryb wyłączności powiedzie się serwer wysyła pozytywną odpowiedź, po czym przechodzi w stan transakcji.

Po otwarciu skrzynki pocztowej, serwer przypisuje wiadomościom numery i bada ich długość. Pierwsza wiadomość w skrzynce otrzymuje numer 1, druga - numer 2, itd. Teraz wymienimy polecenia używane w stanie autoryzacji. Nie każda implementacja serwera POP3 musi używać wszystkich tych poleceń.

`USER string`
: Polecenie przyjmuje argument będący nazwą (identyfikatorem) skrzynki pocztowej znajdującej się na hoście, do którego klient jest aktualnie przyłączony. Jeżeli serwer zwróci odpowiedź pozytywną, klient może wysłać polecenie `PASS` w celu zakończenia autoryzacji albo `QUIT` aby przerwać sesję. Jeżeli serwer zwróci negatywną odpowiedź na polecenie `USER`, wówczas klient może ponowić proces autoryzacji albo przerwać sesję.

`PASS string`
: Polecenie `PASS` może być użyte tylko po poleceniu `USER`. Jako argument przyjmuje hasło dostępu do skrzynki pocztowej. Jeśli hasło jest poprawne serwer zwraca odpowiedź pozytywną. Jeśli hasło jest niepoprawne albo nie można przejść do trybu wyłączności serwer zwraca odpowiedź negatywną.

`APOP name digest`
: Jako pierwszy argument polecenia przyjmuje nazwę skrzynki pocztowej. Drugi argument stanowi ciąg będący wynikiem funkcji wyliczania sumy kontrolnej MD5 dla wyrażenia złożonego z hasła i pewnych niepowtarzalnych wartości. Te niepowtarzalne wartości to identyfikator procesu i czas na hoście przesyłane do klienta w momencie uzyskania połączenia. Dane te są przesyłane w następującej postaci: `process-ID.clock@hostname`.

1.2.2 Stan transakcji

Stan transakcji następuje po stanie autoryzacji. Występujące w nim polecenia opisane są poniżej.

`STAT`
: Polecenie `STAT` nie przyjmuje żadnego argumentu. W odpowiedzi na nie serwer emituje pozytywną odpowiedź zawierającą informacje o zawartości skrzynki pocztowej. Owa odpowiedź ma odpowiedni format. Po pozytywnym identyfikatorze stanu `+OK` występuje znak spacji a następnie liczba oznaczająca liczbę wiadomości znajdujących się w skrzynce pocztowej. Następnie znajduje się kolejny znak spacji i kolejna liczba oznaczająca tym razem łączną wielkość wszystkich wiadomości w skrzynce. Bardziej rozbudowane implementacje mogą zwracać jeszcze inne, dodatkowe informacje o skrzynce. Należy jeszcze zaznaczyć, że wiadomość, która wcześniej została oznaczona jako skasowana nie jest brana pod uwagę przez polecenie `STAT`.

`LIST [msg]`

Polecenie przyjmuje opcjonalny argument będący numerem wiadomości. Jeżeli argument występuje i serwer POP3 zwraca pozytywną odpowiedź, to zawiera ona informacje o wielkości danej wiadomości. Jeżeli argument nie występuje i serwer POP3 zwraca pozytywną odpowiedź, to jest to odpowiedź wieloliniowa, będąca listą wielkości każdej z wiadomości w skrzynce. Owa lista wyświetlana jest w ustalonym formacie. Każda linia rozpoczyna się numerem wiadomości, następnie występuje znak spacji a po nim wielkość wiadomości o odpowiednim numerze.

Wiadomości, które zostały wcześniej skasowane nie są brane pod uwagę przez polecenie `LIST`.

`RETR msg`

Polecenie przyjmuje jeden argument będący numerem wiadomości, która nie może być oznaczona jako skasowana. Jeśli odpowiedź serwera jest pozytywna, jest ona wieloliniowa i niesie ze sobą treść wiadomości o numerze podanym w argumencie.

`DELE msg`

Polecenie przyjmuje jeden argument będący numerem wiadomości, która nie może być oznaczona jako skasowana. Działanie polecenia polega na oznaczeniu wiadomości, o numerze podanym w argumencie, jako skasowanej. Jakiekolwiek późniejsze próby operacji na takiej wiadomości kończą się błędem. Serwer POP3 usuwa fizycznie wiadomości dopiero po przejściu w stan aktualizacji.

`NOOP`

Polecenie nie przyjmuje żadnego argumentu. Polecenie nie powoduje żadnej akcji. Serwer POP3 zwraca jedynie pozytywną odpowiedź. Wysyłane jest przez klienta, aby uniknąć rozłączenia z powodu przekroczenia limitu bezczynności.

`RSET`

Polecenie nie przyjmuje żadnego argumentu. Polecenia `RSET` używa się aby anulować zaznaczenie wszelkich wiadomości w skrzynce jako skasowanych. Serwer reaguje wówczas pozytywną odpowiedzią.

1.2.3 Stan aktualizacji

W momencie, gdy klient wysyła polecenie `QUIT` do serwera będącego w stanie transakcji, sesja POP3 przechodzi w stan aktualizacji. Jeśli komenda `QUIT` zostałaby wysłana ze stanu autoryzacji sesja zostałaby zakończona. Jeśli sesja w stanie transakcji zostanie przerwana z powodu innego niż wysłanie polecenie `QUIT`, wówczas serwer nie przechodzi do stanu aktualizacji i żadne z zaznaczonych do skasowania wiadomości nie zostają usunięte.

`QUIT`

Wszystkie wiadomości zaznaczone jako skasowane, są fizycznie usuwane z serwera. Zwalniana jest blokada wyłącznego dostępu. Zamykane jest połączenie TCP pomiędzy serwerem a klientem.

1.2.4 Opcjonalne polecenia protokołu POP3

`TOP msg n`

Polecenie przyjmuje dwa argumenty. Pierwszy z nich to numer wiadomości (nie może to być wiadomość oznaczona jako skasowana), drugi to ilość linii do pobrania. Użycie polecenia powoduje zwrócenie przez serwer pozytywnej, wieloliniowej odpowiedzi, składającej się z tytułu wiadomości, pustej linii i `n` pierwszych linii wiadomości o numerze `msg`. Polecenie może być stosowane jedynie w stanie transakcji. Jeśli ilość linii podana jako argument jest większa od faktycznej ilości linii w wiadomości, wówczas zwracana jest cała wiadomość.

`UIDL [msg]`

Polecenie przyjmuje opcjonalnie jeden argument, będący numerem wiadomości, która nie została wcześniej oznaczona jako skasowana. Polecenie stosuje się jedynie w stanie transakcji. Serwer zwraca unikalny identyfikator danej wiadomości, jeśli jej numer występuje jako argument. Na polecenie bez argumentu serwer zwraca listę unikalnych identyfikatorów wszystkich, nieskasowanych wiadomości znajdujących się w skrzynce pocztowej.

1.3 Przykładowa sesja POP3

Poniżej przedstawiamy przykładowy dialog klienta z serwerem POP3. Do połączenia z serwerem użyty został program `telnet`. Znakiem „>” oznaczono treść poleceń wpisanych z klawiatury.

```
> telnet pop3.poczta.onet.pl 110
   +OK POP3 [213.180.130.20] onet 1.20 server ready
> user tokrob
   +OK User name accepted, password please
> pass abc123
   +OK Mailbox open, 15 messages, new: 15, your primary
   account: tokrob@poczta.onet.pl
> stat
   +OK 15 6874288
> list
   +OK Mailbox scan listing follows
   1 5519
```

```
   2 95148
   3 99621
   4 35081
   5 614524
   6 604043
   7 58764
   8 507433
   9 4886
   10 3055007
   11 717312
   12 96200
   13 917626
   14 36669
   15 26455
   .
> noop
   +OK No-op to you too!
> dele 1
   +OK Message deleted
> rset
   +OK Reset state
> quit
   +OK Sayonara
```

Rozdział 2

Zmiany w serwerze Teapop

Do przeglądania poczty na serwerze stosuje się jeden z dwóch protokołów: POP3 lub IMAP. Wiele z popularnych klienckich programów pocztowych obsługuje oba te protokoły. Należą do nich między innymi: Netscape/Mozilla, Outlook, Eudora, Eclipse. Dostępnych jest również wiele programów realizujących te protokoły po stronie serwera. Do najczęściej stosowanych należą: WU (Washington University) IMAP/POP3, QuickPOP, Cyrus IMAP. Do nielicznych jednak należą serwery POP3 obsługujące konta wirtualne.

Przez konto wirtualne rozumiemy konto służące jedynie do odbierania i wysyłania poczty, bez możliwości logowania się użytkownika na serwer i uruchamiania na nim jakichkolwiek programów. Może się tutaj pojawić pytanie: jaki sens ma stosowanie kont wirtualnych? Otóż w praktyce mamy do czynienia z dużą liczbą użytkowników zainteresowanych wyłącznie możliwością korzystania z poczty elektronicznej. Często są to osoby, które nie posiadają wystarczającej wiedzy do korzystania z pełnego konta shell'owego na serwerze. Musimy jednocześnie pamiętać, że udostępnianie konta shell'owego wiąże się z ryzykiem wejścia do systemu osób niepowołanych, najczęściej poprzez odgadnięcie łatwego hasła zabezpieczającego konto. Uzasadnia to tworzenie wirtualnych kont pocztowych, które są jedynie wydzielonymi na serwerze plikami zawierającymi wiadomości. Straty wynikające z włamania do konta wirtualnego są zdecydowanie mniejsze niż z włamania do pełnego konta systemowego, co mogłoby w skrajnych przypadkach zakończyć się przejęciem przez hakera całego systemu.

Wirtualne konto pocztowe to pojedynczy plik, w którym trzymane są wiadomości użytkownika. Każde wirtualne konto pocztowe musi być oczywiście "opisane" na serwerze. Opisy te stanowią konfigurację wirtualnych kont pocztowych i zawierają między innymi pełny adres e-mail, hasło oraz położenie pliku z wiadomościami.

Program Teapop [10] jest serwerem POP3 posiadającym możliwość obsługi wielu wirtualnych domen. Konfiguracja kont pocztowych w tym programie może być przechowywana w plikach tekstowych, bądź w bazie danych SQL. Dopuszczalnych jest kilka formatów plików tekstowych z hasłami (shadow,

htpasswd), jeśli chodzi o bazę danych może to być MySQL lub Postgres. O ile w przypadku stosowania plików tekstowych wygodniej jest wydzielić osobny plik dla każdej wirtualnej domeny, to w sytuacji, gdy używamy bazy danych wygodniej przechowywać wszystkie opisy kont w jednej tabeli. Niestety program Teapop wymaga zakładania osobnych tabel dla każdej domeny. Powoduje to konieczność utworzenia tabeli w bazie oraz dodania dodatkowego wiersza do pliku konfiguracyjnego serwera Teapop przy dodawaniu nowej domeny. Jednym z celów naszej pracy jest przeprogramowanie serwera Teapop tak, aby umożliwić przechowywanie informacji o wielu różnych domenach w jednej tabeli bazy danych.

Dokonane zmiany w kodzie źródłowym serwera Teapop prezentowane są w postaci patch'a na oryginalny kod. Taki patch uzyskujemy przy pomocy programu `diff`, który wyłapuje różnice pomiędzy dwoma wskazanymi plikami. Aby takie patch'e były czytelne program `diff` był wołany z opcją "-c", która dodaje po trzy wiersze kontekstu przed i po znalezionej różnicy między plikami. Nazwa *patch* bierze się stąd, że uzyskane przy pomocy `diff` pliki można użyć do naniesienia poprawek na kod oryginalny. Służy do tego program `patch`. W przypadku dużych projektów nie musimy przesyłać kompletnego kodu źródłowgo do potencjalnych użytkowników, wystarczy jedynie przesłać naniesione poprawki. Ponadto kontekstowe patch'e pozwalają szybko zorientować się jakie zmiany i w jakim miejscu zostały dokonane. Każdy patch na początku zawiera pełne ścieżki do porównywanych plików oraz daty ich modyfikacji. Dalej możemy wyróżnić fragmenty zawierające opis zmian wraz z podanym numerem wiersza początkowego i końcowego. W opisie zmian używa się następujących symboli:

`!` – wskazuje zmodyfikowane wiersze,

`+` – oznacza dodany wiersz,

`-` – oznacza wiersz usunięty.

Dokonane zmiany w programie Teapop można podzielić na dwie części: zmiany w module komunikacji z bazą danych oraz zmiany dotyczące pliku konfiguracyjnego. Zanim jednak opiszemy wykonane modyfikacje musimy kilka słów powiedzieć o strukturze danych używanej w programie Teapop.

2.1 Struktura danych w programie Teapop

Używana w programie Teapop struktura danych jest dosyć prosta. Można w niej wyróżnić dwie zasadnicze części: strukturę `POP_AUTH_SQL` zawierającą dane z pliku konfiguracyjnego oraz strukturę `POP_INFO`, w której przechowywane są informacje na temat podłączonego do serwera klienta i związanego z nim użytkownika.

W przypadku serwera Teapop mamy do czynienia z konfiguracją samego programu Teapop umieszczoną w pliku `teapop.passwd` z jednej strony oraz z konfiguracją kont pocztowych. Plik `teapop.passwd` zawiera "metainformacje" niezbędne do działania programu. Pojedynczy wiersz tego pliku odpowiada konfiguracji jednej domeny. Wyjątek stanowi wpis zawierający słowo `default` w polu nazwy domeny, wtedy wpis ten dotyczy domen nie opisanych explicite. W każdym takim wierszu konfiguracyjnym określony jest sposób przechowywania danych o kontach pocztowych. Dostępne formaty oraz sposoby ich użycia opisane są w komentarzu zawartym w pliku `teapop.passwd`. Przykładowy plik konfiguracyjny serwera Teapop zamieszczamy na stronie 35.

Najbardziej interesującą nas metodą przechowywania informacji o kontach pocztowych jest baza danych MySQL. Odpowiedni dla tej metody wpis w pliku `teapop.passwd` może wyglądać następująco[1]:

```
default:*:mysql:/export/home1/mail/:0:root:mail:localhost::
       dbISP:mail:ma1l:tMail:username:password:maildrop:domain:
```

Kolejne pola oddzielone są znakiem ":" i mają następujące znaczenie:

`domain` — nazwa domeny, której dotyczy konfiguracja. Wartość `default` oznacza, że konfigurujemy wszystkie możliwe z obsługiwanych domen.

`IP number` — adres IP serwera. Znak „*" oznacza, że program Teapop nasłuchuje na wszystkich adresach IP systemu.

`authinfo` — metoda przechowywania informacji o kontach pocztowych. W naszym przykładzie jest to baza MySQL.

`maildir` — pełna (bezwzględna) ścieżka (prefix) do miejsca w systemie plików, gdzie znajdują się skrzynki pocztowe wirtualnych użytkowników.

`hash level` — w przypadku dużej ilości skrzynek pocztowych grupuje się je ze względu na początkowe litery alfabetu i umieszcza w odpowiednich katalogach. Przy wartości 1 konta `jacek@domena.pl` oraz `jan@domena.pl` będą umieszczone w katalogu o nazwie `j`. Wartość 0 oznacza brak takich podkatalogów.

`puid` — identyfikator użytkownika systemowego z prawami którego ma działać serwer Teapop. W naszym przykładzie jest to użytkownik root.

`pgid` — identyfikator grupy systemowej z prawami której ma działać serwer Teapop. W naszym przypadku jest to grupa mail.

`phostname` — nazwa serwera na którym znajduje się baza danych MySQL. Wartość `localhost` oznacza, że jest to ten sam komputer na którym działa serwer Teapop.

[1] Z powodów technicznych przykładowy wiersz został złamany.

`pport` — numer portu TCP używany przez bazę danych. Pominięta wartość w naszym wypadku oznacza port standardowy, czyli 3306.

`pdatabase` — nazwa bazy danych w której zebrane są dane o kontach pocztowych.

`pdbuser` — nazwa użytkownika bazy danych uprawnionego do wykonywania zapytań typu SELECT na bazie danych.

`pdbpass` — hasło użytkownika określonego powyżej.

`ptable` — nazwa tabeli w podanej wcześniej bazie danych.

`puserrow` — nazwa kolumny zawierającej nazwy użytkowników.

`ppassrow` — nazwa kolumny zawierającej hasła użytkowników.

`pmailrow` — nazwa kolumny zawierającej względne ścieżki (ustalane względem wartości pola `maildir`) do wirtualnych skrzynek pocztowych.

`pdomainrow` — nazwa kolumny zawierającej nazwy domen.

Powyższy przykład uwzględnia zmiany przez nas wykonane, a mianowicie dodatkowe pole `pdomainrow`.

Poszczególne wiersze pliku `teapop.passwd` wczytywane są do struktur `POP_AUTH_SQL`. Deklarację takiej struktury mamy przedstawioną poniżej.

```
typedef struct _pop_auth_sql {
        char host[BIGSTRING];
        char *port;
        char db[SMALLSTRING];
        char username[SMALLSTRING];
        char password[SMALLSTRING];
        char table[SMALLSTRING];
        char userrow[SMALLSTRING];
        char passrow[SMALLSTRING];
        char mailrow[SMALLSTRING];
        char domainrow[SMALLSTRING];
}POP_AUTH_SQL;
```

Widać, że kolejne pola w tej strukturze odpowiadają wpisom w pliku `teapop.passwd`. Przedstawiona powyżej deklaracja zawiera dodane przez nas pole `domainrow`.

Drugą ważną strukturą danych używaną w programie Teapop jest przedstawiona poniżej struktura `POP_INFO`.

```
typedef struct _pop_info {
        int insck;
        int outsck;
        int autodelete;
        int ignoreimap;
```

```
        int timeout;
        int locktimeout;
        int nodns;
        int useuidl;
        int locktrack;
        int mboxperm;
        int expire;
        int ssl;
        int softlock;
        unsigned short localport;
        char drachost[BIGSTRING];
        char apopstr[BIGSTRING];
        char userid[BIGSTRING];
        char domain[BIGSTRING];
        char maildrop[BIGSTRING];
        char dotlock[BIGSTRING+20];
        char mboxtype;          /* 0 = mbox, 1 = Maildir */
        char chroot[BIGSTRING];
        char localip[40];
        char remoteip[40];
        char remotehost[BIGSTRING];
        FILE *lock;
        FILE *mbox;
        FILE *out;
        POP_MSG *firstmsg;
        POP_AUTH *firstauth;
        void *smask;
}POP_INFO;
```

Inicjowana jest ona w momencie, gdy podłącza się klient POP3. Zawiera dane o połączeniu, takie jak: adres IP klienta, nazwę i domenę użytkownika. W przypadku kont systemowych domena jest taka jak domena serwera na którym założono konta. Użytkownik systemowy logując się w protokole POP3 wystarczy, że poda swój identyfikator. Sytuacja komplikuje się, gdy w grę wchodzą konta wirtualne. Tutaj użytkownik musi podać zarówno identyfikator jak i domenę. Są one zapisywane w polach odpowiednio `userid` i `domain`. Tak więc w przypadku kont wirtualnych użytkownik zobowiązany jest podać pełny adres e-mail podczas logowania. Pole `maildrop` zawiera względną ścieżkę do pliku skrzynki pocztowej natomiast `chroot` to ustalony w konfiguracji Teapop stały prefix do kont pocztowych.

2.2 Zmiany w module komunikacji z bazą danych

Generalnie zmiany w module komunikacji z bazą danych polegają na dodaniu nowej kolumny do tabeli opisującej konta wirtualne i dostosowaniu zapytań, tak aby brana była ona pod uwagę.

Podane przez użytkownika podczas logowania identyfikator i domena bę-

dą przekazane jako fragment zapytania do bazy danych. Dla bezpieczeństwa przed wstawieniem ich do zapytania znaki specjalne w nich występujące (takie jak: apostrof, cudzysłów,itp.) powinny zostać ochronione przy pomocy funkcji `mysql_real_escape_string()`. Funkcja ta tworzy w pamięci kopię podanego napisu z ochronionymi znakami specjalnymi. Aby móc zapamiętać kopię nazwy domeny potrzebujemy dodatkową zmienną `edomain`. Poniższy fragment patch'a opisuje odpowiednią modyfikację.

```
*** teapop-0.3.8/teapop/pop_mysql.c 2003-08-04 17:33:49.000000000 +0200
--- teapop-0.3.8-new/teapop/pop_mysql.c 2004-05-07 12:27:10.519155000 +0200
***************
*** 136,142 ****
  {
    POP_AUTH_SQL     *psql;
    MD5_CTX          ctx;
!   char     *ptr, buf2[512], *ppass, md5pass[32], euserid[BIGSTRING*2+1];
    int      counter;
    unsigned char digest[16];
    MYSQL_RES *result;
--- 136,142 ----
  {
    POP_AUTH_SQL     *psql;
    MD5_CTX          ctx;
!   char     *ptr, buf2[512], *ppass, md5pass[32], euserid[BIGSTRING*2+1],
                                                        edomain[BIGSTRING*2+1];
    int      counter;
    unsigned char digest[16];
    MYSQL_RES *result;
```

Teraz możemy zawołać funkcję `mysql_real_escape_string()`, aby utworzyć `edomain`. Ważniejsza jednak jest zmiana w zapytaniu SELECT do bazy danych. Należało to zapytanie zmodyfikować tak, aby porównywana była nazwa domeny podana przy logowaniu z domeną znajdującą się w bazie. Obie zmiany są przedstawione poniżej.

```
***************
*** 164,172 ****
    /*======================*/
    mysql_real_escape_string(mysql, euserid, pinfo->userid,
       strlen(pinfo->userid));
!   sprintf(mybuf, "SELECT %s, %s%s%s FROM %s WHERE %s = '%s'",
       psql->userrow, psql->passrow, (*(psql->mailrow) == NULL ? "" : ", "),
!      psql->mailrow, psql->table, psql->userrow, euserid);

    len = strlen(psql->userrow);
    if (check_host) {
--- 164,174 ----
    /*======================*/
    mysql_real_escape_string(mysql, euserid, pinfo->userid,
       strlen(pinfo->userid));
!   mysql_real_escape_string(mysql, edomain, pinfo->domain,
!      strlen(pinfo->domain));
!   sprintf(mybuf, "SELECT %s, %s%s%s FROM %s WHERE %s = '%s' AND %s = '%s'",
       psql->userrow, psql->passrow, (*(psql->mailrow) == NULL ? "" : ", "),
!      psql->mailrow, psql->table, psql->userrow, euserid, psql->domainrow,
                                                              edomain);

    len = strlen(psql->userrow);
    if (check_host) {
```

Następną poważną zmianą w tym module było zadbanie o prawidłowe wyliczenie pełnej ścieżki do pliku zawierającego wirtualną skrzynkę pocztową. Ścieżka ta składa się ze stałego prefixu określonego w `teapop.passwd` oraz ze względnej ścieżki `pinfo->maildrop`. Wartość `pinfo->maildrop` wyliczana jest następująco: jeśli w konfiguracji danego konta podano nazwę pliku to jest ona używana, w przeciwnym razie brana jest nazwa domeny (jako nazwa podkatalogu) i nazwa użytkownika (jako nazwa pliku).

O ile wcześniej nie była uwzględniana domena przy określaniu pełnej ścieżki do pliku skrzynki pocztowej, a jedynie nazwa użytkownika to teraz używane są obie wartości, czyli `pinfo->domain` i `pinfo->userid`.

```
***************
*** 242,253 ****
                if (ptr[0] != '\0')
                        strncpy(pinfo->maildrop, ptr,
                            sizeof(pinfo->maildrop));
!               else
!                       strncpy(pinfo->maildrop, pinfo->userid,
!                           sizeof(pinfo->maildrop));
          } else {
!               strncpy(pinfo->maildrop, pinfo->userid,
!                   sizeof(pinfo->maildrop));
        }
        return (0);
 }
--- 244,256 ----
                if (ptr[0] != '\0')
                        strncpy(pinfo->maildrop, ptr,
                            sizeof(pinfo->maildrop));
!               else {
!                       snprintf(pinfo->maildrop, sizeof(pinfo->maildrop),
!                                "%s/%s", pinfo->domain, pinfo->userid);
!                 }
        } else {
!               snprintf(pinfo->maildrop, sizeof(pinfo->maildrop),
!                         "%s/%s", pinfo->domain, pinfo->userid);
        }
        return (0);
 }
```

2.3 Zmiany w pliku konfiguracyjnym

Zmiany w tej części kodu źródłowego sprowadzają się do wzbogacenia pliku konfiguracyjnego `teapop.passwd` o dodatkowe pole określające nazwę kolumny zawierającej nazwy domen. Pole to nazwaliśmy `pdomainrow`.

Pierwsza część zmian polega na dodaniu nowej zmiennej `pdomainrow`, w której przechowywana będzie wartość odpowiedniego pola z `teapop.passwd`.

```
*** teapop-0.3.8/teapop/pop_passwd.c 2003-11-26 15:34:53.000000000 +0100
--- teapop-0.3.8-new/teapop/pop_passwd.c 2004-05-07 12:20:09.856828000 +0200
***************
*** 99,105 ****
        char *pfile, *pmax;                     /* Used for TEXTFILE */
```

```
        char *phostname, *pport, *pdatabase;    /* Used for SQLs */
        char *pdbuser, *pdbpass, *ptable;       /*  ditto  */
!       char *puserrow, *ppassrow, *pmailrow;   /*  ditto  */
  #ifdef HAVE_LDAP
        char *prootdn, *pbind;                  /* Used for LDAPAUTH */
  #endif
--- 99,105 ----
        char *pfile, *pmax;                     /* Used for TEXTFILE */
        char *phostname, *pport, *pdatabase;    /* Used for SQLs */
        char *pdbuser, *pdbpass, *ptable;       /*  ditto  */
!       char *puserrow, *ppassrow, *pmailrow, *pdomainrow;      /*  ditto  */
  #ifdef HAVE_LDAP
        char *prootdn, *pbind;                  /* Used for LDAPAUTH */
  #endif
```

Następna część modyfikacji to przeczytanie z pliku `teapop.passwd` nazwy kolumny i podstawienie jej do zmiennej `pdomainrow`. Jeśli nazwy tej nie podano zgłaszany będzie błąd w logu systemowym.

```
***************
*** 534,539 ****
--- 534,548 ----
                        }
                        *ptr++ = '\0';

+                       /* 17th arg is name of the row with domain info */
+                       pdomainrow = ptr;
+                       if ((ptr = strchr(pdomainrow, ':')) == NULL) {
+                               syslog(LOG_ERR, "line %d corrupt; please "
+                                   "check the syntax", line);
+                               continue;
+                       }
+                       *ptr++ = '\0';
+
                        psql = malloc(sizeof(POP_AUTH_SQL));
                        if (psql == NULL) {
                                memset(tmpauth, 0, sizeof(POP_AUTH));
```

Na koniec wartość lokalnej zmiennej `pdomainrow` kopiujemy do zmiennej o charakterze globalnym `psql->domainrow`. Struktura wskazywana przez `psql` zawiera przeczytaną właśnie z `teapop.passwd` konfigurację. Wskaźnik `psql` zostaje zapamiętany w strukturze `POP_INFO` związanej z podłączonym klientem.

```
***************
*** 563,568 ****
--- 572,579 ----
                            sizeof(psql->passrow) - 1);
                        strncpy(psql->mailrow, pmailrow,
                            sizeof(psql->mailrow) - 1);
+                       strncpy(psql->domainrow, pdomainrow,
+                           sizeof(psql->domainrow) - 1);
                        tmpauth->extra = psql;
                } else if ( 0 ||
  #ifdef HAVE_JAVA
```

Rozdział 3

Programy administracyjne

W konfiguracja wirtualnych skrzynek pocztowych wydzielić można dwie części: pierwsza dotyczy konfiguracji serwera pocztowego, w naszym wypadku Sendmail zaś druga dotyczy konfiguracji serwera POP3, w naszym wypadku Teapop. Jest to podyktowane tym, że korzystanie z poczty wymaga dwóch odrębnych programów po stronie serwera. Jednym jest serwer SMTP, służący do rozsyłania poczty, drugim jest serwer POP3 lub IMAP, służący do przeglądania poczty.

W ramach pracy dyplomowej napisanej w ubiegłym roku, zatytułowanej „System zarządzania pocztowymi hostami wirtualnymi na serwerze ISP" został zaprojektowany system kont wirtualnych oraz zostały napisane programy do konfiguracji serwera SMTP. Jeśli natomiast chodzi o konfigurację tych kont od strony serwera POP3 została wykonana aplikacja internetowa do zdalnego zarządzania kontami. W praktyce okazało się, że ta aplikacja jest w pewnych sytuacjach niewystarczająca. Ponieważ jest interaktywna, nie pozwala na automatyzację podstawowych czynności jak: zakładanie, usuwanie i modyfikacja kont wirtualnych. Bardzo często wygodniejszy jest zestaw skryptów wykonujących te zadania. Przygotowanie zestawu takich skryptów jest jednym z celów niniejszej pracy. Wszystkie skrypty zostały napisane w języku Perl, z uwagi na to, że posiada on wygodny interfejs DBI do bazy danych SQL, a w takiej bazie właśnie przechowywane są dane o kontach wirtualnych serwera Teapop.

Do konfiguracji serwera Teapop napisane zostały programy: `popuseradd`, `popuserdel`, `popusermod`, `poppasswd` oraz `poplistusers`. Wykonują one wyłącznie zmiany w bazie MySQL, z której korzysta serwer Teapop. Nie mają one żadnego związku z konfiguracją serwera Sendmail. Do konfiguracji Sendmail'a służą skrypty: `virtuseradd` oraz `virtuserdel`. Dla wygody administratora zostały napisane również dwa skrypty: `mailuseradd` i `mailuserdel`, które nie robią nic innego jak tylko wywołują odpowiednio `virtuseradd`, `popuseradd` i `poppasswd` oraz `virtuserdel` i `popuserdel`, z odpowiednimi parametrami. Być może ta ilość skryptów powoduje zamieszanie, ale w ten sposób możemy niezależnie konfigurować serwer Sendmail i Teapop, co może być w pewnych sytuacjach pożądane.

Programy były testowane na komputerze theta w Instytucie Matematyki UwB.

3.1 Konfiguracja kont POP3 w MySQL

Aby lepiej zrozumieć funkcjonowanie całego systemu kont wirtualnych jak i samych skryptów, podajemy opis tabeli z bazy danych MySQL wykorzystywanej przez serwer Teapop. Zgromadzone są w niej informacje o kontach wirtualnych POP3.

```
+------------+--------------+------+-----+------------+----------------+
| Field      | Type         | Null | Key | Default    | Extra          |
+------------+--------------+------+-----+------------+----------------+
| ID         | int(11)      |      | PRI | NULL       | auto_increment |
| TS         | timestamp(14)| YES  |     | NULL       |                |
| CT         | timestamp(14)| YES  |     | NULL       |                |
| username   | varchar(16)  |      | MUL |            |                |
| domain     | varchar(128) |      |     |            |                |
| password   | varchar(16)  |      |     |            |                |
| maildrop   | varchar(16)  | YES  |     | NULL       |                |
| active     | enum('Y','N')| YES  |     | Y          |                |
| start_date | date         |      |     | 0000-00-00 |                |
| expire_date| date         |      |     | 0000-00-00 |                |
+------------+--------------+------+-----+------------+----------------+
```

Poszczególne kolumny tej tabeli mają następujące znaczenie:

`ID` — unikalny identyfikator rekordu w tabeli,

`TS` — czas ostatniej modyfikacji rekordu,

`CT` — czas utworzenia rekordu,

`username` — nazwa konta użytkownika,

`domain` — nazwa domeny,

`password` — hasło użytkownika,

`maildrop` — względna ścieżka do skrzynki pocztowej użytkownika, liczona względem konfiguracji serwera Teapop,

`active` — pole decydujące o tym, czy dane konto jest aktywne (Y), czy też zablokowane (N),

`start_date` — od kiedy konto jest aktywne,

`expire_date` — do kiedy konto jest aktywne.

Kolumny `start_date` i `expire_date` mogą być wykorzystane w zastosowaniach komercyjnych, gdzie często ma miejsce wykupienie skrzynki pocztowej na określony czas. Natomiast zmieniając wartość pola `active` można łatwo zablokować skrzynkę użytkownika, na przykład w przypadku nadużycia.

Przykładowa zawartość takiej tabeli podana jest poniżej:

ID	username	domain	password	maildrop	active	start_date	expire_date
1	kowalski	poczta.pl	ab3z4hnHA5WdU	NULL	N	2001-11-11	2008-04-20
2	burski	poczta.pl		NULL	Y	2004-04-20	2008-04-20
3	kowalski	domena.pl		NULL	Y	2004-04-20	2008-04-20
4	smith	poczta.us	abGFm1Rd/Y6Lc	NULL	N	2004-04-20	2008-04-20
5	john	mail.com		/export	N	2001-11-11	2008-04-20
6	ania	gazeta.pl	abJTIvuctg9SU	NULL	Y	2001-11-11	2008-04-20
7	robert	teges.pl	abJTIvuctg9SU	NULL	Y	2001-11-11	2001-11-11
8	gosia	domena.com		NULL	Y	2001-02-01	2008-04-20
9	goisia	domena.pl		NULL	Y	2222-02-02	2005-02-01
10	rtok	local	abGFm1Rd/Y6Lc		Y	2004-04-23	2008-04-23

Przedstawione dalej w tekście skrypty manipulują zawartością powyższej tabeli.

3.2 Założenie wirtualnej skrzynki pocztowej POP3

Do założenia wirtualnej skrzynki POP3 służy skrypt popuseradd.

```
popuseradd username@domain.tld [-m plik]
      [-a Y/N] [-s data] [-e data]
```

Aby dodać nową skrzynkę, jako argument podaje się pełny adres e-mail, czyli nazwę użytkownika i domenę w standardowej postaci `username@domain.tld`, na przykład:

```
popuseradd kowalski@poczta.pl
```

Nazwa użytkownika zapisywana jest w kolumnie `username`, natomiast nazwa domeny w kolumnie `domain`.

Pełny adres jako argument jest obowiązkowy i nie może być pominięty. Opcjonalnie przy zakładaniu nowej skrzynki pocztowej możemy używać dodatkowych przełączników:

`-m plik`
: Podana wartość `plik` jest zapisywana w kolumnie `maildrop`. Domyślnie skrzynka wirtualna jest umieszczana w pliku do którego pełna ścieżka utworzona jest przez standardowy prefix, domenę i nazwę użytkownika, na przykład:

```
/export/home1/mail/poczta.pl/kowalski
```

gdzie `/export/home1/mail` jest tym właśnie standardowym prefixem. Przy pomocy opcji `-m` możemy zmienić lokalizację pliku skrzynki użytkownika. Jeśli `popuseradd` wywołamy w następujący sposób:

```
popuseradd kowalski@poczta.pl -m kowalski_inbox
```

wówczas pełna ścieżka do pliku skrzynki tworzonego użytkownika będzie jak poniżej:

```
/export/home1/mail/kowalski_inbox
```

Pamiętajmy o tym, że program `popuseradd` sam nie tworzy żadnych plików wpisuje tylko odpowiednie informacje do bazy danych.

`-a Y|N`
: Przełącznik -a służy do ustawienia skrzynki pocztowej na aktywną jeśli podamy `Y`, bądź nieaktywną przy wartości `N`. Podana wartość zapisywana jest w bazie w kolumnie `active`. Przykład użycia przełącznika:

```
popuseradd kowalski@domena.pl -a Y
```

`-s data`
: Czasami istnieje potrzeba ustalenia daty od której skrzynka pocztowa zaczyna funkcjonować. W tym celu używamy właśnie przełącznika `-s`. Przełącznik ten wymaga argumentu, którym, jest pełna data, od której skrzynka zaczyna działać. Data podawana jest w formacie MySQL: rok-miesiąc-dzień. Przykład użycia przełącznika `-s`:

```
popuseradd kowalski@poczta.pl -s 2004-06-16
```

Gdy nie użyjemy tej opcji, wartość kolumny `start_date` ustawiana jest na aktualną datę systemową.

`-e data`
: Ten przełącznik ustala datę wygaśnięcia skrzynki pocztowej. Podana data zapisywana jest w kolumnie `expire_date`. Co do formatu daty obowiązują te same zasady co przy opcji `-s`. Przykładowe użycie:

```
popuseradd kowalski@poczta.pl -e 2007-06-16
```

Jeśli przełącznik jest pominięty to data wygaśnięcia skrzynki pocztowej powstanie poprzez dodanie 4 lat do daty początkowej, z kolumny `start_date`.

Kod źródłowy skryptu zamieszczono na str. 24.

3.3 Usunięcie wirtualnej skrzynki pocztowej POP3

Do usunięcia wirtualnej skrzynki pocztowej służy skrypt `popuserdel`.

```
popuserdel username@domain.tld
```

Skrypt pobiera jeden argument, którym jest adres e-mail, na przykład:

```
popuserdel kowalski@poczta.pl
```

Skrypt usuwa z bazy danych cały wiersz dotyczący podanej skrzynki wirtualnej POP3. Nie są usuwane jakiekolwiek pliki.

Kod źródłowy skryptu zamieszczono na str. 26.

3.4 Modyfikacja wirtualnej skrzynki pocztowej POP3

Skrypt `popusermod` służy do edycji konfiguracji wirtualnej skrzynki pocztowej POP3.

```
popusermod username@domain.tld [-m plik]
      [-a Y/N] [-s data] [-e data]
```

Jako obowiązkowy argument skrypt pobiera pełny adres e-mail. Można też stosować opcjonalne przełączniki, których znaczenie jest takie samo jak w skrypcie `popuseradd`.

Kod źródłowy skryptu zamieszczono na str. 29.

3.5 Zmiana hasła wirtualnej skrzynki pocztowej POP3

Zmianę hasła do wirtualnej skrzynki pocztowej wykonujemy przy pomocy skryptu `poppasswd`.

```
poppasswd username@domain.tld
```

Wywołując ten skrypt podajemy pełny adres e-mail, na przykład:

```
poppasswd kowalski@poczta.pl
```

Skrypt poprosi o podanie nowego hasła. Hasło przed zapisaniem do bazy danych szyfrowane jest przy pomocy systemowej funkcji `crypt()`. Skrypt ten powinien być uruchamiany tylko przez administratora, ponieważ nie jest wykonywa żadna autoryzacja przed zmianą hasła. Nie jest tak jak w przypadku polecenia `passwd` zmieniającego hasło konta systemowego, gdzie przed zmianą hasła użytkownik pytany jest o hasło aktualne.

Kod źródłowy skryptu zamieszczono na str. 27.

3.6 Przeglądanie listy wirtualnych skrzynek pocztowych POP3

Do przeglądania konfiguracji wirtualnych skrzynek pocztowych został napisany skrypt `poplistusers`.

```
maillistusers [polecenie [domena]]
```

Przy uruchomieniu tego skryptu bez dodatkowych parametrów zostanie wyświetlona pełna lista wirtualnych skrzynek pocztowych. Na wynikowej liście jako pierwszy podany jest adres e-mail, a za nim względna ścieżka do pliku skrzynki wirtualnej, o ile taka została wprowadzona do bazy.

Parametr `domena` zawęża wyszukiwanie do określonej domeny. Parametr `polecenie` może przyjmować następujące wartości:

`all`
: Wyświetla wszystkie adresy e-mail. Jeżeli nie podana jest nazwa domeny to komenda ta może zostać opuszczona.

`valid`
: Wyświetla te adresy e-mail, które odpowiadają działającym wirtualnym skrzynkom pocztowym, to znaczy takim, które są aktywne i nie wygasły.

`invalid`
: Wyświetla te adresy e-mail, które nie spełniają warunków komendy `valid`.

`active`
: Wyświetla aktywne adresy e-mail.

`inactive`
: Wyświetla nieaktywne adresy e-mail.

`expired`
: Wyświetla te adresy e-mail, które wygasły. Konto uważa się za wygasłe, gdy jego data rozpoczęcia funkcjonowania jest późniejsza niż bieżąca data lub gdy data wygaśnięcia jest wcześniejsza niż data bieżąca.

`unexpired`
: Wyświetla te adresy e-mail, które nie wygasły, czyli te, które spełniają negację warunku opisującego komendę `expired`.

Kod źródłowy skryptu zamieszczono na str. 31.

3.7 Założenie wirtualnej skrzynki pocztowej SMTP i POP3

Skrypt `mailuseradd` jest prostym skryptem shell'owym. Jego uruchomienie powoduje dodanie wirtualnego konta pocztowego zarówno do konfiguracji serwera SMTP jak i serwera POP3.

```
mailuseradd username@domain.tld [-m plik]
      [-a Y/N] [-s data] [-e data]
```

Skrypt wywołuje kolejno `virtuseradd` (skrypt z pracy dyplomowej napisanej w ubiegłym roku), `popuseradd` oraz `poppasswd` z odpowiednimi argumentami. Składnia wywołania skryptu `mailuseradd` jest taka jak skryptu `popuseradd`.

Kod źródłowy skryptu zamieszczono na str. 33.

3.8 Usunięcie wirtualnej skrzynki pocztowej SMTP i POP3

Skrypt `mailuserdel` wywołuje skrypty `virtuserdel` i `popuserdel`.

```
mailuserdel username@domain.tld
```

Uruchamia się go podając jako obowiązkowy argument pełny adres e-mail usuwanego konta pocztowego.

Kod źródłowy skryptu zamieszczono na str. 34.

Dodatek A

Skrypty

popuseradd

```
#!/opt/cfw/bin/perl
#
# Add a virtual account to POP3 configuration
#
# Last modified: May 12, 2004
#
# Copyright (c) 2004 Robert Tokarski & Mariusz Zynel.
#
# This software is FREE. You can use and/or redistribute it for any
# purpose in either, modified, or unmodified form, under the terms of the
# GNU General Public License as published by the Free Software Foundation.
#
# The above copyright notice and this permission notice shall be included
# in all copies or substantial portions of this software.
#
# THIS SOFTWARE IS PROVIDED AS IS AND COME WITH NO WARRANTY OF ANY KIND,
# EITHER EXPRESSED OR IMPLIED. IN NO EVENT WILL THE COPYRIGHT HOLDER BE
# LIABLE FOR ANY DAMAGES RESULTING FROM THE USE OF THIS SOFTWARE.

use strict;
use DBI;
use Getopt::Std;

my ($username, $domain, $maildrop, $active, $start, $expire);

our ($opt_m, $opt_a, $opt_s, $opt_e);


(my $self = $0) =~ s/.*\///;

sub usage{
    die "Usage: $self username\@domain.tld" .
        "[-m maildrop] [-a Y/N] [-s date] [-e date]\n";
}

if ($#ARGV < 0) {
  usage();
}

@ARGV[0] =~ m/([^\@]+)\@(.*)/;

my $username = $1;
my $domain = $2;
```

```
shift @ARGV;

getopt('mase');

if ($opt_m) {
    if ($opt_m =~ /^[*]$/) {
        $maildrop = $opt_m;
    } else {
        usage();
    }
} else {
    $maildrop = "";
}

if ($opt_a) {
    if ($opt_a =~ /^[YN]$/) {
        $active = $opt_a;
    } else {
        usage();
    }
} else {
  $active = "Y";
}

if ($opt_s) {
    if ($opt_s =~ /^[0-9]{4}-[0-9]{2}-[0-9]{2}$/) {
        $start = "'$opt_s'";
    } else {
        usage();
    }
} else {
    $start = "now()";
}

if ($opt_e) {
    if ($opt_e =~ /^[0-9]{4}-[0-9]{2}-[0-9]{2}$/) {
        $expire = "'$opt_e'";
    } else {
        usage();
    }
} else {
    $expire = "date_add(start_date, interval 4 year)";
}

# Connect to the database
my $dbh = DBI->connect("DBI:mysql:host=localhost:database=dbISP;",
                                                        "mail", "mail");

# Update database
$dbh->do("insert into tMail values(null, null, null, '$username', " .
          "'$domain', '', '$maildrop', '$active', $start, $expire) ")
                  || die "ERROR: MySQL: $Mysql::db_errstr";

# Disconnect
$dbh->disconnect();

system("mailpasswd $username\@$domain");
```

popuserdel

```
#!/opt/cfw/bin/perl
#
# Delete virtual account from POP3 configuration
#
# Last modified: May 12, 2004
#
# Copyright (c) 2004 Robert Tokarski & Mariusz Zynel.
#
# This software is FREE. You can use and/or redistribute it for any
# purpose in either, modified, or unmodified form, under the terms of the
# GNU General Public License as published by the Free Software Foundation.
#
# The above copyright notice and this permission notice shall be included
# in all copies or substantial portions of this software.
#
# THIS SOFTWARE IS PROVIDED AS IS AND COME WITH NO WARRANTY OF ANY KIND,
# EITHER EXPRESSED OR IMPLIED. IN NO EVENT WILL THE COPYRIGHT HOLDER BE
# LIABLE FOR ANY DAMAGES RESULTING FROM THE USE OF THIS SOFTWARE.

use strict;
use DBI;

(my $self = $0) =~ s/.*\///;

sub usage {
    die "Usage: $self username\@domain.tld\n";
}

@ARGV[0] =~ m/([^\@]+)\@(.*)/;

my $username = $1;
my $domain = $2;

if ($username eq "" || $domain eq "") {
    usage();
} else {
    # Connect to the database
    my $dbh = DBI->connect("DBI:mysql:host=localhost:database=dbISP;",
                                                        "mail", "mail");

    # Update database
    $dbh->do("delete from tMail where username='$username' and " .
          "domain = '$domain'") || die "ERROR: MySQL: $Mysql::db_errstr";

    # Disconnect
    $dbh->disconnect();
}
```

poppasswd

```
#!/opt/cfw/bin/perl
#
# Change password to a virtual account
#
# Last modified: May 12, 2004
#
# Copyright (c) 2004 Robert Tokarski & Mariusz Zynel.
#
# This software is FREE. You can use and/or redistribute it for any
# purpose in either, modified, or unmodified form, under the terms of the
# GNU General Public License as published by the Free Software Foundation.
#
# The above copyright notice and this permission notice shall be included
# in all copies or substantial portions of this software.
#
# THIS SOFTWARE IS PROVIDED AS IS AND COME WITH NO WARRANTY OF ANY KIND,
# EITHER EXPRESSED OR IMPLIED. IN NO EVENT WILL THE COPYRIGHT HOLDER BE
# LIABLE FOR ANY DAMAGES RESULTING FROM THE USE OF THIS SOFTWARE.

use strict;
use DBI;

(my $self = $0) =~ s/.*\///;

sub getPasswd {
    my $msg = shift;
    print "$msg";
    system('/usr/bin/stty', '-echo');  # Disable echoing
    my $passwd = <STDIN>;
    system('/usr/bin/stty', 'echo');   # Turn it back on
    chomp $passwd;
    return $passwd;
}

sub usage {
    die "Usage: $self username\@domain.tld\n";
}

if ($#ARGV+1 != 1) {
    usage;
}

@ARGV[0] =~ m/([^\@]+)\@(.*)/;

my $username = $1;
my $domain = $2;

if ($username eq "" || $domain eq "") {
    usage;
} else {
    my $password = getPasswd("Enter new password:\n");
    my $rpassword = getPasswd("Renter the password:\n");

    if ($password eq $rpassword) {

        if (length($password) < 6 || $password !~ /[^a-z].*[^a-z]/) {
            die "ERROR: password is trivial\n";
        }

        if (length($password) > 8) {
            die "ERROR: password is too long, max 8 characters\n";
        }
```

```
        $password = crypt($password, "ab");

        # Connect to the database
        $dbh = DBI->connect("DBI:mysql:host=localhost:database=dbISP;",
                                                        "mail", "mail");

        my $dbh = DBI->connect("DBI:mysql:host=localhost:database=dbISP;",
                                                        "mail", "mail");

        # Update database
        my $rows_affected = $dbh->do("update tMail set " .
            "password = '$password' where username = '$username' and " .
              "domain = '$domain'") || die "ERROR: MySQL: $Mysql::db_errstr";

        if ($rows_affected <= 0) {
            print "ERROR: $username\@$domain not found\n";
        }

        # Disconnect
        $dbh->disconnect();
    } else {
        print "ERROR: entered passwords do not match\n";
    }
}
```

popusermod

```
#!/opt/cfw/bin/perl
#
# Modify attributes of a virtual account in POP3 configuration
#
# Last modified: May 12, 2004
#
# Copyright (c) 2004 Robert Tokarski & Mariusz Zynel.
#
# This software is FREE. You can use and/or redistribute it for any
# purpose in either, modified, or unmodified form, under the terms of the
# GNU General Public License as published by the Free Software Foundation.
#
# The above copyright notice and this permission notice shall be included
# in all copies or substantial portions of this software.
#
# THIS SOFTWARE IS PROVIDED AS IS AND COME WITH NO WARRANTY OF ANY KIND,
# EITHER EXPRESSED OR IMPLIED. IN NO EVENT WILL THE COPYRIGHT HOLDER BE
# LIABLE FOR ANY DAMAGES RESULTING FROM THE USE OF THIS SOFTWARE.

use strict;
use DBI;
use Getopt::Std;

my ($username, $domain, $maildrop, $active, $start, $expire);

our ($opt_m, $opt_a, $opt_s, $opt_e);

(my $self = $0) =~ s/.*\///;

sub usage{
    die "Usage: $self username\@domain.tld [-m maildrop]" .
        "[-a Y/N] [-s date] [-e date]\n";
}

if ($#ARGV < 0) {
    usage();
}

@ARGV[0] =~ m/([^\@]+)\@(.*)/;

my $username = $1;
my $domain = $2;

shift @ARGV;

getopt('mase');

if ($opt_m) {
    $maildrop = "'$opt_m'";
} else {
    $maildrop = "maildrop";
}

if ($opt_a) {
    if ($opt_a =~ /^[YN]$/) {
        $active = "'$opt_a'";
    } else {
        usage();
    }
} else {
    $active = "active";
}
```

```
if ($opt_s) {
    if ($opt_s =~ /^[0-9]{4}-[0-9]{2}-[0-9]{2}$/) {
        $start = "'$opt_s'";
    } else {
        usage();
    }
} else {
    $start = "start_date";
}

if ($opt_e) {
    if ($opt_e =~ /^[0-9]{4}-[0-9]{2}-[0-9]{2}$/) {
        $expire = "'$opt_e'";
    } else {
        usage();
    }
} else {
    $expire = "expire_date";
}

# Connect to the database
my $dbh = DBI->connect("DBI:mysql:host=localhost:database=dbISP;",
                                                        "mail", "mail");

# Update database
$dbh->do("update tMail set maildrop = $maildrop, active = $active," .
    "start_date = $start, expire_date = $expire " .
        "where username = '$username' and domain = '$domain'")
            || die "ERROR: MySQL: $Mysql::db_errstr";

# Disconnect
$dbh->disconnect();
```

poplistusers

```
#!/opt/cfw/bin/perl
#
# List virtal accounts and their atributes from POP3 configuration
#
# Last modified: May 12, 2004
#
# Copyright (c) 2004 Robert Tokarski & Mariusz Zynel.
#
# This software is FREE. You can use and/or redistribute it for any
# purpose in either, modified, or unmodified form, under the terms of the
# GNU General Public License as published by the Free Software Foundation.
#
# The above copyright notice and this permission notice shall be included
# in all copies or substantial portions of this software.
#
# THIS SOFTWARE IS PROVIDED AS IS AND COME WITH NO WARRANTY OF ANY KIND,
# EITHER EXPRESSED OR IMPLIED. IN NO EVENT WILL THE COPYRIGHT HOLDER BE
# LIABLE FOR ANY DAMAGES RESULTING FROM THE USE OF THIS SOFTWARE.

use strict;
use DBI;

my ($dbh, $sth, $record);
my ($username, $domain, $query, $command);

our ($opt_a, $opt_i, $opt_u, $opt_e);

(my $self = $0) =~ s/.*\///;

my %commands = (
    "all", "null is null",
    "valid", "active='Y' and start_date <= now() and now() < expire_date",
    "invalid", "active='N' or now() < start_date or expire_date < now()",
    "active","active='Y'",
    "inactive","active='N'",
    "expired","now() < start_date or expire_date < now()",
    "unexpired","start_date <= now() and now() < expire_date");

sub usage{
    print("Usage: $self [command [domain]]\n\n");
    print("command is one of:\n");
    print("  all\t\tall mailboxes (default)\n");
    print("  valid\tmailbox is active and unexpired\n");
    print("  invalid\tmailbox is inactive or expired\n");
    print("  active\tmailbox is active\n");
    print("  inactive\tmailbox is not active\n");
    print("  expired\tmailbox expired\n");
    print("  unexpired\tmailbox did not expire\n");
    exit 1;
}

if ($#ARGV == -1) {
    $query = '';
} elsif ($#ARGV == 0) {
    $command = @ARGV[0];
} elsif ($#ARGV == 1) {
    $command = @ARGV[0];
    $domain = @ARGV[1];
} else {
    usage();
}

$query = $commands{$command};
```

```
if ($command && $query eq "") {
    usage();
}

if ($query) {
    $query = "where $query";
}

if ($domain) {
    $query = "$query and domain = '$domain'";
}

# Connect to the database
$dbh = DBI->connect("DBI:mysql:host=localhost:database=dbISP;",
                                                    "mail", "mail");

# Print the contents
    $sth = $dbh->prepare("select username, domain, maildrop " .
                    "from tMail $query")
                        || die "ERROR: MySQL: $Mysql::db_errstr";
    $sth->execute();
    while ($record = $sth->fetchrow_hashref) {
        printf("%s@%s\t%s\n", $record->{username},
                            $record->{domain}, $record->{maildrop});
    }

# Disconnect
    $dbh->disconnect();
```

mailuseradd

```
#!/bin/sh
#
# Add a virtual account to both SMTP and POP3 configuration
#
# Last modified: Jun 4, 2004
#
# Copyright (c) 2004 Robert Tokarski & Mariusz Zynel.
#
# This software is FREE. You can use and/or redistribute it for any
# purpose in either, modified, or unmodified form, under the terms of the
# GNU General Public License as published by the Free Software Foundation.
#
# The above copyright notice and this permission notice shall be included
# in all copies or substantial portions of this software.
#
# THIS SOFTWARE IS PROVIDED AS IS AND COME WITH NO WARRANTY OF ANY KIND,
# EITHER EXPRESSED OR IMPLIED. IN NO EVENT WILL THE COPYRIGHT HOLDER BE
# LIABLE FOR ANY DAMAGES RESULTING FROM THE USE OF THIS SOFTWARE.

SELF=`basename $0`

PATH=/usr/bin:/usr/sbin:/opt/cfw/sbin
export PATH

if [ $# -lt 1 ]; then
  echo "Usage: $SELF username@domain.tld [-m maildrop] " .
       "[-a Y/N] [-s date] [-e date]"
  exit 1;
fi

virtuseradd $1
popuseradd $*
poppasswd $1
```

mailuserdel

```
#!/bin/sh
#
# Remove a virtual account from both SMTP and POP3 configuration
#
# Last modified: Jun 4, 2004
#
# Copyright (c) 2004 Robert Tokarski & Mariusz Zynel.
#
# This software is FREE. You can use and/or redistribute it for any
# purpose in either, modified, or unmodified form, under the terms of the
# GNU General Public License as published by the Free Software Foundation.
#
# The above copyright notice and this permission notice shall be included
# in all copies or substantial portions of this software.
#
# THIS SOFTWARE IS PROVIDED AS IS AND COME WITH NO WARRANTY OF ANY KIND,
# EITHER EXPRESSED OR IMPLIED. IN NO EVENT WILL THE COPYRIGHT HOLDER BE
# LIABLE FOR ANY DAMAGES RESULTING FROM THE USE OF THIS SOFTWARE.

SELF=`basename $0`

PATH=/usr/bin:/usr/sbin:/opt/cfw/sbin
export PATH

if [ $# -ne 1 ]; then
  echo "Usage: $SELF username@domain.tld"
  exit 1;
fi

virtuserdel $1
popuserdel $1
```

teapop.passwd

Przykładowy plik konfiguracyjny serwera Teapop po dodaniu nowego pola `pdomainrow`.

```
# $Id: //depot/Teapop/0.3/etc/teapop.passwd.in#3 $
#
# Info for where to get the passwords
#
# Syntax:
#
# <domain>:<IP number>:<authinfo>:
#
# NOTE: All lines end with a colon ':'.
#
# Where <domain> is what the user types in after the @-sign, or one of the
# following strings, which have special meanings.
#
# empty    - When the user doesn't enter anything after the @-sign, or has no
#            @-sign at all.
# default  - This entry will be used if no other entry matches.
#
#
# The <IP number> is used for handling of virtual domains on a server. If
# there are no virtual domains and all users connect to one domain on the
# server, this field can be left blank, but it is preferable if you add a *,
# indicating it should cover all domains.
#
#
# The <authinfo> can be one of the following:
#
# reject
#          - Rejects all auths for this domain.
# passwd:<mailspool>:<hash level>:
#        - Will authenticate against the systems' useraccounts.
# textfile:<maildir>:<hash level>:<uid>:<gid>:<passwordfile>:<max accounts>:
#        - Will read the first <max account> accounts in <passwordfile> and
#          then attempt to validate the user against those records. If Teapop
#          finds a valid record for the user it will chroot to <maildir>
#          drop privs to <uid> and <gid>. After that it will open the mailbox
#          specified for the user in <passwordfile> and of course hashed
#          with the specified <hash level>.
#          The format of <passwordfile> is:
#          <userid>:<password>:<mailbox>
# htpasswd:<maildir>:<hash level>:<uid>:<gid>:<passwordfile>:<max accounts>:
#        - Exactly the same as 'textfile', with the exception of the
#          format of the <passwordfile>. It's an Apache plain-text password
#          file, created with htpasswd using the system crypt().
# pgsql:<maildir>:<hash level>:<puid>:<pgid>:<phostname>:<pport>:<pdatabase>:
   <pdbuser>:<pdbpass>:<ptable>:<puserrow>:<ppassrow>:<pmailrow>:<pdomainrow>
# mysql:<maildir>:<hash level>:<puid>:<pgid>:<phostname>:<pport>:<pdatabase>:
   <pdbuser>:<pdbpass>:<ptable>:<puserrow>:<ppassrow>:<pmailrow>:<pdomainrow>
#        - puid = drop teapop's privs to uid
#          pgid = drop teapop's privs to gid
#          phostname = connect to a sql server on this server
#          pport = connect to this port (blank = standard port)
#          pdatabase = use this database
#          pdbuser = user to login to the database as
#          pdbpass = password to login to the database
#          ptable = table info about accounts can be found in
#          puserrow = the row that contains usernames
#          ppassrow = row that contains passwords
#          pmailrow = row that contains name of the maildrop, if empty,
#                     asume the maildropname is the same as the username
#                     (this is relative maildir)
#          pdomainrow = row that contains (virtual) domain name, this makes
```

```
#                       possible to store all passwords in a single table
#
# Maildir support: If you want to use Maildir's instead of mbox, please
#                  make sure that the mailbox name for users ends with
#                  a slash (/).
# mbox support:    Make sure the users mailbox name do NOT end with
#                  a slash (/).
#
# examples:
#
#  If user connects without a domain part, authenticate against the
#  accounts on the system. The mailboxes are in the users homedir and are
#  called .Mailbox
#
#      empty:*:passwd:~/.Mailbox:0:
#
# To handle virtual domains add the IP number of the domain the user is
# connecting to, as in the following example. If you add the IP number then
# the client will connect to the virtual domain with that IP number. It can
# be left blank or a * can be used to allow authorization on all IP numbers
# on a machine.
#
#     example.com:192.168.1.125:passwd:/var/mail:1:
#     or
#     empty:192.168.1.125:passwd:/var/mail:1:
#
# If a user logs in with example.com as domain, authenticate against the
# systems' accounts. The mailboxes are in /var/mail, with one hash level,
# and have the same name as the useraccount, ie /var/mail/u/userid
#
#     example.com:*:passwd:/var/mail:1:
#
# If the user logs in with schmoop.com, authenticate against a textfile
# called /home/susan/passwd. Before opening a mailbox, drop privs to
# uid susan and gid susan and then chroot to /home/susan/mail. Also,
# susan is only allowed to have 10 accounts for schmoop.com.
#
#     schmoop.com:*:textfile:/home/mail:0:susan:susan:/var/teapop/passwd:10:
#
# For all others, autoreject them
#
#     default:*:reject:
#
#
# Short example to get you going at once:
#empty:*:passwd:/var/mail::
#default:*:reject
default:*:mysql:/export/home1/mail/:0:root:mail:localhost::dbISP:mail:mail:
    tMail:username:password:maildrop:domain:
```

Bibliografia

[1] Alligator D., Bunce T., *Perl DBI: programowanie*, O'Reilly, 2000.

[2] Kernighan B.W., Ritchie D.M., *Język C*, Wydawnictwa Naukowo-Techniczne, 1987.

[3] Yarger R.J., Reese G., King T., *MySQL i mSQL*, O'Reilly, 1999.

[4] Wall L., *Programowanie Perl*, 2nd Edition, O'Reilly, 2000.

[5] Post Office Protocol - Version 2,
`http://www.faqs.org/rfcs/rfc937.html`

[6] Post Office Protocol - Version 3,
`http://www.faqs.org/rfcs/rfc1939.html`
`http://www.faqs.org/rfcs/rfc1725.html`

[7] Qmail,
`http://www.qmail.org`

[8] Sendmail,
`http://www.sendmail.org`

[9] Simple Mail Transfer Protocol,
`http://www.faqs.org/rfcs/rfc821.html`

[10] Teapop,
`http://www.toontown.org/teapop/`