Filtr protokołu SMTP

Praca inżynierska

Zbigniew Artemiuk
Z.Artemiuk@stud.elka.pw.edu.pl

11 sierpnia 2008

Spis treści

1 Wstęp

Obecnie w Internecie, pomimo szeregu rozwijających się form komunikacji, chociażby technologii takich jak tekstowa komunikacja czasu rzeczywistego (czyli IM - instant messaging i popularne chaty) czy też jeszcze bardziej zaawansowanych technologii, które umożliwiają przesyłanie głosu oraz wideo (chociażby Skype), pospolite maile cały czas znajdują zastosowanie. Używamy ich chyba obecnie najczęściej do kontaktów biznesowych, rodzinnych, przyjacielskich, głównie w celach informacyjnych (czyli sam tekst), jak również do przekazywania niewielkich plików. Chyba każdy internauta legitymuje się przynajmniej jedną skrzynką mailową (a czasami jest ich znacznie więcej). Każdy spotkał się także z niechcianą pocztą (niechcianymi mailami), nazywaną bardzo ogólnym określeniem Spam, i narzędziami (chociażby tymi wbudowanymi w klientów pocztowych dostępnych z poziomu przeglądarki internetowej), które umożliwiają odseparowanie takiej poczty, od tej która niesie interesującą nas zawartość merytoryczną. Separacja ta to tzw. filtrowanie ze względu na obiekt filtracji zwane filtracja poczty. Dokonanie filtracji wymaga jednak poznania dokładnie w jaki sposób transportowane są nasze maile, a to wszystko zawarte jest w protokole Simple Mail Transfer Protocol (w skrócie SMTP). Coś tutaj jeszcze dopisać o protokole, o jego wadach, o konieczności filtracji.

2 Cel i zakres pracy

Celem pracy jest zaprojektowanie oprogramowania, które po uruchomieniu umożliwiałoby filtrację wiadomości w ramach protokołu SMTP. Oprogramowanie to byłoby konfigurowalne poprzez plik tekstowy zawierający reguły dotyczące wiadomości SMTP i zachowania oprogramowania po napotkaniu takiej wiadomości. Reguły te dawałyby możliwość następującej filtracji:

- wykrywanie wiadomości posiadających bądź nie posiadających określony nagłówek (nagłówki)
- wykrywanie wiadomości posiadających bądź nie posiadających określonej frazy w nagłówku (nagłówkach)
- wykrywanie wiadomości posiadających bądź nie posiadających określonej frazy w nagłówkach części maila (gdy nie jest to prosty mail)
- wykrywanie określonych content-type-ow w wiadomościach
- wykrywanie wiadomości o określonej wielkości części (jeżeli wiadomość jest jednoczęściowa to tyczy się treści wiadomości)

Po natrafieniu na wiadomość spełniającą kryteria danej reguły zostanie ona w zależności od zapisu w konfiguracji:

- odrzucona (wyfiltrowana)
- przepuszczona, a jej obecność zostanie zaznaczona w logach
- przepuszczona

3 Protokół SMTP

3.1 Historia powstania

3.1.1 ARPANET czyli początki Internetu

Historia powstania protokołu SMTP jest ściśle związana z początkami Internetu. Internet zaś i jego kreowanie związane jest bezpośrednio ze swoim przodkiem czyli ARPANETEm.

ARPANET (Advanced Research Projects Agency Network) został stworzony przez przez jedną z agencji United States Department of Defense (departament bezpieczeństwa Stanów Zdjednoczonych) o nazwie ARPA (Advanced Research Projects Agency). Nazwa agencji została później przekształcona na DARPA (D od Defence). Agencja ta miała zając się rozwojem nowych technologii na potrzebę amerykańskiego wojska.

W miarę wchodzenia w życie komputerów wykorzystywanych w ramach agencji powstała idea stworzenia sieci pomiędzy nimi, która to umożliwiłaby komunikacje pomiędzy ich użytkownikami. Idea ta została po raz pierwszy zaproponowana przez Josepha Carla Robnetta Licklidera z firmy Bolt, Beranek and Newman (obecnie BBN Technologies) w sierpniu 1962 w serii notatek na temat koncepcji "Międzygalaktycznej Sieci Komputerowej". Zawierała ona prawie wszystko czego możemy doświadczyć w dzisiejszym Internecie.

W październiku 1963 roku Licklider został mianowany szefem programu Behavioral Sciences and Command and Control w ARPA. Przekonał on wtedy Ivana Sutherlanda i Boba Taylora, że jego wizja jest czymś naprawdę istotnym. Sam Licklider nie doczekał jednak żadnych konkretnych prac w kierunku jej urzeczywistnienia, gdyż opuścił ARPA.

ARPA i Taylor cały czas byli zainteresowanie stworzeniem sieci komputerowej, ażeby zapewnić naukowcom pracującym w ramach ARPA w różnych lokalizacjach, dostęp do innych komputerów, które firma oferowała. Istotne było także, aby nowe oprogramowanie i rezultaty badań były jak najszybciej widoczne dla każdego użytkownika sieci. Sam Taylor posiadał 3 różne terminale, które dawały mu podłączenie do 3 różnych komputerów - jeden do SDC Q-32 w Santa Monica, drugi w ramach projektu Project Genie do komputera na Uniwersytecie w Kalifornii (Berkley) i ostatni do komputera z Multicsem w MIT (The Massachusetts Institute of Technology).

Taylor w taki sposób opowiadał od podłączeniu do tych komputerów: " Dla każdego z tych terminali miałem inny zestaw poleceń. Dlatego też kiedy rozmawiałem z kimś z Santa Monica, a poźniej chciałem ten sam temat skonsultować z kimś z Berkley albo MIT, musiałem przesiąść się do innego terminala. Oczywistym wtedy wydało mi się, że musi być 1 terminal, który obsłuży te 3 połączenia. Idea ta to właśnie ARPANET"

Do połowy 1968 roku kompletny plan sieci został stworzony i po zatwierdzeniu przez ARPA, zapytanie ofertowe RFQ (Request For Quotation) zostało posłane do 140 potencjalnych wykonawców. Większość potraktowała propozycję jako dziwaczną. Tylko 12 firm złożyło oferty z czego 4 zostały uznane za najważniejsze. Do końca roku wyłoniono 2 firmy, z których ostatecznie 7 kwietnia 1969 roku została wybrana firma BBN.

Propozycja BBN była najbliższa planom ARPA. Pomysłem ich było stworzenie sieci z mały komputerów zwanych Interface Message Processors (bardziej

znanych jako IMPs), które to obecnie nazywamy routerami. IMPsy z każdej strony zapewniały funkcje przechowywania i przekazywania pakietów, a połączone były między sobą przy użyciu modemów podpiętych do łączy dzierżawionych (o przepustowości 50 kbit/sekundę). Komputery podłączone były do IMPsów poprzez specjalny bitowy interfejs. W ten sposób stawały się one częścią sieci ARPANET.

Do zbudowania pierwszej generacji IMPsów BBN wykorzystała komputer Honeywell DDP-516. Został on wyposażony w 24kB pamięci rdzenia (z możliwością rozszerzenia) oraz 16 kanałów Direct Multiplex Control (DMC) do bezpośredniogo dostępu do tej pamięci. Poprzez DMC podłączane były komputery użytkowników (hosty) i modemy. Dodatkowo 516 otrzymał ezstaw 24 lamp, które pokazywały status kanałów komunikacyjnych IMPa. Do każdego IMPa można było podłączyć do czterech hostów i mógł się on komunikować z 6 zdalnymi IMPami poprzez współdzielone łącza.

Zespół z BBN (początkowo 7 osób) szybko stworzył pierwsze działające jednoski (IMPy). Cały system, który zawierał zarówno sprzęt jak i pierwsze oprogramowania zarządzające pakietami, został zaprojektowany i zainstalowany w ciągu 9 miesięcy.

Początkowo ARPANET składał się z 4 IMPów. Zostały one zainstalowane w:

- UCLA (University of California, Los Angeles), gdzie Leonard Kleinrock założył centrum pomiaru sieci (Network Measurment Center)
- The Stanford Research Institute's Augmentation Research Center, gdzie Douglas Engelbert stworzył system NLS, który między innymi wprowadził pojęcie hypertextu
- UC Santa Barbara
- The University of Utah's Graphics Department, gdzie przebywał ówcześnie Ivan Sutherland

3.1.2 Piersze wiadomości w ARPANETcie

Pierwsza komunikacja host-host w sieci ARPANET wykorzystywała protokół 1822, który definiował sposób w jakis host przesyłał wiadomość do IMPa. Format wiadomości był tak zaprojektowany, żeby bez problemu mógł pracować z szerokim zakresem architektur. Zasadniczo wiadomość składała się z:

- typu wiadomości
- adresu hosta
- pola z danymi

W celu wysłania wiadomości do innego hosta, host wysyłający powinien sformatować wiadomość tak, aby ta zawierała adres hosta docelowego oraz dane, a następnie dokonać transmisji wiadomości przez interfejs sprzętowy 1822. IMP dostrzeże dostarczenie wiadomości albo poprzez dostarczenie jej bezpośrednio do hosta docelowego albo poprzez przekazanie jej do kolejnego IMPa. Kiedy wiadomość została odebrana przez docelowego hosta, IMP do którego host był

podłączony wysyła potwierdzenie odbioru (zwane Ready for Next Message or RFNM) do hosta wysyłającego.

W przeciwieństwie do obecnych protokołów datagramowych w Interecie (takich jak no IP), ARPANETowy porotokół 1822 zapewniał niezawodność w taki sposób, że informował o niedostarczonej wiadomości. Niemniej protokół 1822 nie był odpowiedni do żonglowania wieloma połączeniami w różnych aplikacjach uruchomionych na pojedynczym hoście. Problem ten został rozwiązany dzięki wprowadzeniu na hostach Network Control Program (NCP), dzięki któremu możliwe było niezawodne, z kontrołą przepływu, dwukierunkowe połączenia pomiędzy różnymi procesami na różnych hostach. NCP implementował kolejną warstwę znajdującą się na górze stosu protokołów. Dzięki niemu aplikacje, które miały mieć już jakąś konkretną funkcjonalność, mogły wykorzystywać spójny interfejs i korzystać swobodnie z dobrodziejstw ARPANETu czyli wykonywać połączenia do innych aplikacji przez sieć.

3.1.3 SNDMSG i READMAIL

Na początku roku 1970, powstał program (a w zasadzie 2 oddzielne) do wysyłania i odbierania wiadomości. Zaimplementował go Ray Tomlinson. Programy te to SNDMSG i READMAIL. Pierwsza wersja tych programów, była kolejną implementacją, która umożliwiała wymiane informacji między użytkownikami jednej maszyny (jednego hosta), a konkretnie komponowanie, adresowanie i wysyłanie wiadomości do skrzynek użytkowników. Już jednak w 1971 Tomlinson stworzył pierwszą aplikację ARPANETową (w ramach prac nad systemem TENEX), która umożliwiała wysyłanie wiadomości do dowolnych hostów. Tomlinson dokonał usprawnień w programie SNDMSG przy okazji pracy nad projektem CPYNET, którego celem było stworzenie protokołu do wymiany plików między komputerami w sieci.

Skrzynkę emailową był wówczas był zwykły plik o określonej nazwie. Specjalną właściwością jego było to, że miał ochronę taką, że umożliwiał innym użytkownikom dopisywanie do pliku. Mogli oni więc pozostawiać kolejne wiadomości ale nie mogli ich czytać ani nadpisywać wcześniejszych (jedynie właściciel mógł to robić). Tomilson zauważył, że CPYNET może dopisywać zawartość do pliku skrzynki mailowej, na takiej zasadzie jak SNDMSG (SNDMSG umiał zapisywać wtedy tylko na lokalnej maszynie). Dlatego też SNDMSG mógł po prostu skorzystać z kodu CPYNET i bezpośrednio dostarczać wiadomości poprzez połączenie sieciowe do zdalnych skrzynek poprzez dopisywanie kolejnych informacji do plików na innych hostach.

Brakującą częścią była możliwość dopisywania do plików przy pomocy CPYNET. Dotychczas mógł on tylko słać i odbierać pliki. Dodanie tej funkcjonalności nie było czymś wielkim dla twórców protokołu i funkcjonalność ta wkrótce zainstniała.

Następnie Tomilson włączył kod CPYNET do SNDMSG. Pozostało jedynie rozróżnienie maili lokalnych od maili zdalnych. Dlatego też Tomilson zdecydował się, że maile zdalne będą rozpoznawane po tym, że po loginie (czyli wyznaczniku użytkownika do którego wiadomość jest słana) nastąpi specjalny znaczek @ (ang. at, a polska znana wszystkim małpa) , a tuż po nim nazwa hosta czyli zdalnego komputera, na którym skrzynkę ma dany loginem użytkownik. Tomilson tak powiedział o wyborze małpy jako znaczka rozdzielającego login od hosta: "I am frequently asked why I chose the at sign, but the at sign just makes

sense" (w wolnym tłumaczeniu: Jestem często pytany czemu wybrałem znaczek małpy, ale on po prostu miał sens).

Pierwsza wiadomość została przesłana pomiędzy maszynami, które fizycznie były obok siebie. Jedyne połączenie jakie było między maszynami (oprócz podłogi ;]) było za pośrednictwem sieci ARPANET. Tomilson przesłał wiele wiadomości do samego siebie z jednej maszyny na drugą. Pierwsze treści wiadomości od razu zostały zapomniane. Najprawdopodobniejsza ich treść była w stylu QWERTYUIOP. Kiedy Tomilson był zadowolony z programu wysłał do swoich kolegów z zespołu wiadomość z instrukcją jak słać wiadomości przez sieć. I tak pierwsza wysłana wiadomość ogłosiła swoje istnienie.

Te pierwsze wiadomości zostały wysłane pod koniec 1971 roku. Następne wydanie TENEXa, które ukazało się w 1972 roku, zawierało SNDMSG, z możliwością wysyłania maili przy użyciu sieci ARPANET. Protokół CPYNET wkrótce został zastąpiony prawdziwym protokołem do transferu plików i posiadał specyficzne dodatki do obsługi maila.

3.1.4 MAIL i MLFL w protokole FTP

Protokołem, o którym wcześniej była mowa był protokół FTP, którego specyfikacja wstępna została zawarta w RFC (Request For Comment) 114. Kolejne ulepszenia protokołu zostały dokonana w roku 1972 i wtedy też weszły do niego nowe polecenia pozwalające korzystać ze skrzynek emailowych. Poleceniami tymi były:

- MAIL - polecenie to pozwalało użytkownikowi przy pomocy telnetu wysłać maila. Pomocne to było gdy użytkownik ten nie korzystał ze swojego hosta i logował sie do niego poprzez protokól telnet
- MLFL - polecenie to pozwalało na normalne skonstruowanie maila i przesłanie odpowiednio wskazanego pliku jako jego treści

Protokół ten stał się aż do roku 1980 standardem przesyłania emaili w ARPANETcie. Wtedy to został wypart przez protokół SMTP, który z pewnymi usprawnieniami funkcjonuje aż do dziś.

3.1.5 Rozwój innych programów do obsługi poczty

Zanim jednak powstał protokół SMTP cały czas pracowano nad rozwojem ówczesnych programów chociażby do odbioru maili. W ten sposób na prośbe Steve'a Lukasika Lawrence Roberts, ówczesny dyrektor IPTO, stworzył program RD, który składał się z makr w edytorze TECO (Text Editor and COrrector).

Program RD umożliwiał:

- sortowanie emaili po temacie i dacie
- czytanie, zapisywanie i czytanie wiadomości w dowolnym porządku

RD nie powstał więc jako wynik badań, ale z czysto praktycznej potrzeby swobodnego zarządzania emailami.

Wkrótce powstały też kolejne ulepszenia programu RD oraz SENDMSG, takie jak NRD czy WRD, które to wprowadzały kolejne ulepszenia istniejących już implementacji.

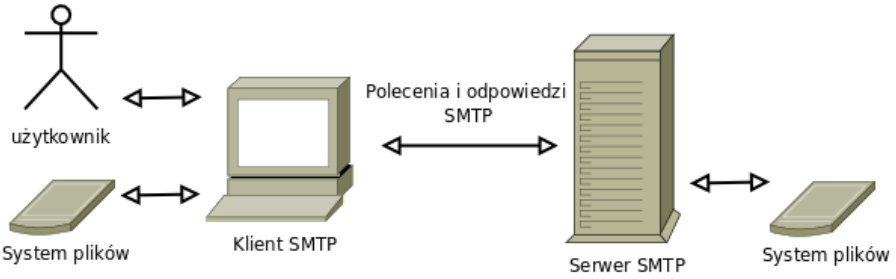
Warte wspomnienia jest także powstanie programu MSG, który umożliwiał między innymi przekazywanie maili (ang. forwarding), konfigurowalny interfejs, czy też polecenie Odpowiedz (ang. Answer), które automatycznie uzupełniało adres odbiorcy. MSG można było nazwać pierwszym nowoczesnym programem pocztowym.

W 1977 roku różne formaty wiadomości zostały zebrane w jedną spójną specyfikację co doprowadziło do stworzenia RFC 733. Specyfikacja ta połączyła wcześniej istniejące dokumentację z odrobioną innowacji i była jednocześnie pierwszym RFC, które deklarowało standard Internetowy (ówcześnie ARPANETowy).

3.2 Szczegóły protokołu

W 1982 roku powstało główne RFC opisujące protokół SMTP - RFC 821. W 2001 roku zostało ono rozbudowane i powstał dokument, który jest obowiązującą obecnie specyfikacją protokołu. Informacje te zawarte zostały w RFC o numerze 2821.

3.2.1 Model protokołu, podstawy



Komunikacja w protokole SMTP oparta jest na nastepującym modelu: kiedy klient SMTP posiada wiadomość, którą chce przetransmitować ustanawia dwukierunkowy kanał transmisyjny z serwerem SMTP. Klient SMTP bierze na siebie odpowiedzialność za przetransportowanie wiadomości do jednego lub wielu serwerów SMTP, lub zgłoszenie błędu jeśli operacji przekazania maila nie powiodła się.

Adres serwera SMTP klient determinuje poprzez zamianę domeny podanej w adresie na pośredni host (Mail eXchanger) lub ostateczny host docelowy.

Serwer SMTP może być ostatecznym celem albo też pośrednim "przekaźnikiem" (po odebraniu wiadomości to on może przejąć rolę klienta SMTP) albo "bramą" (może transportować wiadomość dalej używając innego protokołu niż SMTP). Polecenia protokołu SMTP generowane są przez klienta i posyłane do serwera. Odpowiedzi słane są jak reakcja na żądania klienta.

Innymi słowy wiadomość może być przetransportowana w trakcie pojedynczego połączenia pomiędzy pierwotnym adresatem, a ostatecznym klientem albo może składać się z kilku pośrednich połączeń. W obu przypadkach zachowana jest odpowiedzialność za wiadomość - protokół wymaga od serwera wzięcia odpowiedzialności za dostarczenie wiadomości albo w przypadku nieprawidłowości za zgłoszenie błędu.

Kiedy kanał transmisyjny zostanie zestawiony klient SMTP inicjuje transakcję maila. Składa się ona z serii poleceń, które mają na celu wyspecyfikowanie

nadającego i celu wiadomości, a następnie przesłania jej zawartości (oczywiście z nagłówkami, które wchodzą w jej skład). Kiedy ta sama wiadomość jest adresowana do wielu odbiorców, protokół przesyła jedną kopię wiadomość dla wszystkich odbiorców w obrębie jednego hosta.

Serwer reaguję na każdą komendę odpowiedziami. Wskazują one akceptację komendy (tej przesłanej od klienta) i oczekiwanie na kolejne, bądź też wystąpienie tymczasowych albo stałych błędów.

Kiedy mail zostanie przetransmitowany, klient może zamknąć połączniee albo też zainicjować kolejną tranzakcję innego maila. Dodatkowo klienty SMTP mogą używać połączenia z serwerem w celach np. weryfikacji adresu email bądź też uzyskanie adresów subskrybentów listy dyskusyjnej.

Jak zostało wcześniej zasugerowane transmisja w protokole może odbywać się bezpośrednio pomiędzy hostem ślącym wiadomośc, a hostem docelowym, kiedy są one połączone tym samym serwisem transportowym. Kiedy tak nie jest, transmisja odbywa się poprzez hosty pośrednie. Hosty te wybierane są poprzez mechanizm serwera domen (DNS) zwany Mail eXchanger.

3.2.2 Model rozszerzeń

W wyniku prac, które rozpoczęły się około dekadę po wydaniu pierwszego RFC dotyczącego protokołu SMTP (czyli RFC 822), protokół został rozszerzony i pojawiły się w nim nowe funckjonalności. Standardowy model został więc zmodyfikowany o dodatkowe serwisy. Pozwalają one na ustalenie pomiędzy klientem, a serwerem usług dodatkowych jakie są oni w stanie obsłużyć (poza podstawowymi wymaganiami SMTP). Mechanizm rozszerzeń SMTP określa środki, za pomocą których klient i serwer mogą dokonać wzajemnego ropoznania, a serwer może także poinformować klienta o rozszerzeniach, które on wspiera.

Współczesna implementacja SMTP musi obsługiwać podstawowe mechanizmy rozszerzeń. Przykładowo serwer musi wspierać komendę EHLO, jeśli nawet nie implementuje innych specyficznych rozszerzeń, natomiast klient powinien skłaniać się ku używaniu komendy EHLO niż HELO.

SMTP jest szeroko używany i pojawiło się wiele bardzo dobrych implementacji protokołu wyposażonych w liczne rozszerzenia. Społeczność w Internecie jednak uważa, że niektóre z serwisów są bardzo ważne, mimo tego, że nie powstały gdy protokół był po raz pierwszy projektowany. Jeśli serwisy te mają zostać dodane to w zachowaniem zgodności wstecz.

Obecnie szkielet rozszerzeń składa się z:

- Polecenia EHLO, które powinno być używane zamiast wcześniejszego HELO
- Rejestr rozszerzeń SMTP
- Dodatkowe parametry do poleceń SMTP MAIL i RCPT
- Opcjonalnych zastąpień poleceń w protokole SMTP, jak na przykład DATA w transmisji nie przy pomocy ASCII (ang. pipelining opisany później)

Siłą SMTP jest jego prostota. Doświadczenia z protokołami pokazują, że te z kilkoma możliwościami stają się powszechnie używanymi, podczas gdy te z wieloma możliwościami przestają być czytelne.

3.2.3 Sesja protokołu

Sesja jest inicjowana kiedy klient otwiera połączenie do serwera i serwer odpowie wiadomością zapraszającą.

SMTP serwer po podłączeniu się klienta może po kodzie powitalnym 220 poinformować klienta o swoim oprogramowaniu i jego wersji.

220 mail.foo.com SuperSMTP v 6.1.2 Service ready

Protokół SMTP pozwala serwerowi na odmówienie transakcji podczas gdy formalnie umożliwia na podłączenie się do serwera SMTP. Wtedy zamiast kodu 220 serwer odpowiada kodem 554.

554 Transaction failed (Or, in the case of a connection-opening response, "No SMTP service here")

Serwer w tym przypadku musi czekać zanim zakończeny połączenie na komende QUIT ze strony klienta. W trakcie tego oczekiwania wszystkie inne komendy zostaną potraktowane odpowiedzią 503.

503 Bad sequence of commands

Kiedy serwer przywita się z klientem i klient otrzyma wiadomość powitalną, standardowo wysyła on polecenie EHLO do serwera i w wiadomości tej przedstawia się.

EHLO bar.com

Użycie komendy EHLO wskazuje, że klient jest w stanie obsłużyć rozszerzenia protokołu i prosi serwer o ich listę. Starsze systemy SMTP, które nie obsługują rozszerzeń protokołu czy też starsze klienty SMTP, które nie wymagają tychże rozszerzeń, mogą używać komendy HELO.

HELO bar.com

Serwer w odpowiedzi na HELO nie powinien zwrócić listy rozszerzeń. Jeżeli serwer zwróci na EHLO

500 Command not recognized

, wtedy klient powinien być w stanie przesłać komende HELO.

Następnie dochodzi do transakcji maila. Rozpoczyna się ona poleceniem MAIL, która identyfikuje nadawcę. Po niej następuje seria poleceń RCPT, które przekazują informację o odbiorcy, a na sam koniec DATA czyli ciało maila, które kończone jest wyznacznikiem koća maila. Znacznik ten potwierdza transakcję.

Pierwszym krokiem w procedurze transakcji jest komenda MAIL. Ogólna jej składnia wygląda następująco:

MAIL FROM:<reverse-path> [SP <mail-parameters>] <CRLF>

przykładowo

MAIL FROM:<bob@example.org>

Poleceni to mówi odbiorcy SMTP (serwerowi), że rozpoczynana jest nowa transakcja. Dlatego powinien on zresetować (chyba na danym połączeniu?) wszystkie poprzednie stany tabel i buforów, włączając w to odbiorców i dane maila. Argument <reverse-path> zawiera adres źródłowej skrzynki mailowej, która może posłużyć w celu informacji o ewentualnych błędach (będzie dalej opisane [4.2]). Jeśli polecenie zostanie zaakcpetowane serwer odpowiada poleceniem 250 OK.

250 OK

Jeśli przesłany adres nie został zaakceptowany serwer musi odpowiedzieć czy błąd jest stały (zawsze będzie się pojawiał przy tym adresie) czy tymczasowy

(klient będzie mógł wykonać ponowną próbę, która może zakończyć się sukcesem). Czasami jednak serwer może dokonać akceptacji adresu przekazanego w poleceniu MAIL i dopiero po analizie adresów przesłanych w poleceniu RCPT, może zgłosić błąd. Normalnie jednak gdy błąd jest stały odpowiedź serwera to

550 mailbox not found

albo tymczasowy

503 i co tutaj?

Historycznie <reverse-path> może nie tylko zawierać adres skrzynki, jednakże obecne oprogramowanie nie powinno używać routingu źródła.

Parametry opcjonalne w poleceniu MAIL <mail-parameters> służą do negocjacji rozszerzeń protokołu i będą opisane później. (2.2 sekcja - to juz pisalem - moze gdzies dalej będzie napisane co dokładnie mogą tutaj gadać)

Następnym krokiem w transakcji jest polecenie RCPT. Jego składnia to

RCPT TO:<forward-path> [SP <rcpt-parameters>] <CRLF>

Argumentem tego polecenia jest adres (z reguły adres skrzynki oraz domena) identyfikujący odbiorcę. Jeśli zostanie on zaakceptowany przez serwer odpowiada on komunikatem 250 OK i zapisuje sobie argument forward-path. Jeżeli adres nie jest prawidłowy serwer odpowiada kodem 550. Z reguły odpowiedź wygląda tak

550 No such user

Procedura ta może być powtórzona parokrotnie przykładowo

C: MAIL FROM:<bob@example.org> S: 250 Ok C: RCPT TO:<alice@example.com> S: 250 Ok

W tym przypadku kopia maila trafi do 2 użytkowników korzystających z tego samego serwera pocztowego (tej samej domeny). Jak już wcześniej było wspomniane dzieje się to przy nawiązaniu 1 połączenia.

Argument <forward-path> może zawierać więcej niż jeden adres skrzynki odbiorczej. Ze względów historycznych mogą się tutaj pojawić również lista hostów źródłowych i skrzynka odbiorcza, jednak klienty SMTP (jak już wcześniej było wspomniane) nie powinny korzystać z tej możliwości. Serwer jednak musi być przygotowany na taką ewentualność, ale powinien zignorować tą listę.

Dodatkowe parametry (<rcpt-parameters>) będą omówione później.

Trzecim krokiem w trakcie transakcji jest polecenie DATA. Składnia jego jest prosta po prostu

DATA <CRLF>

Jeżeli polecenie zostało zaakceptowane serwer SMTP zwraca status 354 i prosi klienta aby odpowiedział jak najszybciej treścią maila, kończąc transmisję znacznikiem końca maila. Przykładowo

354 End data with <CR><LF>.<CR><LF>

Kiedy serwer otrzyma znak końca wiadomości, a mail zostanie zapisany odbiorca maila odpowiada standardowo 250 OK. Koniec wiadomości klient zaznacza poprzez przesłanie linii zawierającej jeden znak "." (kropkę). Występują dodatkowo odpowiednia procedura zabezpieczająca nieporozumienie związane z tym, że przy przekazywaniu tekstu wiadomości klient prześle linie zawierającą jedynie znak kropki "." (4.5.2 sekcja).

Polecenie DATA może nie udać się tylko w 2 przypadkach:

- Jeżeli nie było polecenia MAIL albo RCPT albo oba zostały odrzucone wtedy serwer na DATA możę odpowiedzieć

  503 Command out of sequence

albo też
554 No valid recipients
Jeżeli któraś z tych odpowiedzi (albo inna z serii 5yz czyli mówiących o błędzie) zostanie odebrana przez klienta, nie powinien on przesyłać treści wiadomości. Powinien on robić to tylko w przypadku odpowiedzi 354.

- Jeżeli polecenie DATA zostało zaakceptowane odpowiedzią 354, może dojść do nieudanego przesłania maila tylko gdy transakcja jest niekompletna (brak zdefiniowancyh odbiorców), albo niespodziewanie odbiorca przestanie być dostępny lub też serwer z innych powodów (np. polityka ochrony) serwer postanowi odrzucić wiadomość.

W praktyce jednak niektóre serwery nie dokonują weryfikacji odbiorcy przed otrzymaniem całej wiadomości. Nie jest to dobre zachowanie, gdyż jeżeli po zaakceptowaniu adresów odbiorców nagle po poleceniu DATA serwer odpowie "550 mailbox not found" to klient nie jest w stanie określić, który adresat jest nieprawidłowy.

Serwer nie powinien na tym etapie odrzucać wiadomości na podstawie nagłówków przekazanych w treści wiadomości (po poleceniu DATA).

Oto cała przykładowa, prawidłowa sesja protokołu SMTP:
Klient otwiera połączenie do serwera.
S: 220 smtp.example.com ESMTP Postfix
C: HELO relay.example.org
S: 250 Hello relay.example.org, I am glad to meet you
C: MAIL FROM:<bob@example.org>
S: 250 Ok
C: RCPT TO:<alice@example.com>
S: 250 Ok
C: RCPT TO:<theboss@example.com>
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: From: "Bob Example«bob@example.org>
C: To: Alice Example <alice@example.com>
C: Cc: theboss@example.com
C: Date: Tue, 15 Jan 2008 16:02:43 -0500
C: Subject: Test message
C:
C: Hello Alice.
C: This is a test message with 5 headers and 4 lines in the body.
C: Your friend,
C: Bob
C: .
S: 250 Ok: queued as 12345
C: QUIT
S: 221 Bye
Serwer zamyka połączenie.

3.2.4 Polecenia do debugowania adresów

SMTP dostarcza poleceń za pomocą których można dokonać weryfikacji nazwy użytkownika albo uzyskać zawartość listy mailingowej. Dokonuje się tego przy pomocy poleceń VRFY i EXPN, których wspacie powinno być zaimplementowane w serwerze SMTP.

Argumentem polecenia VRFY jest po prostu napisa, który składa się z nazwy użytkownika lub też dodatkowo domeny. Jeżeli w wyniku polecenia serwer zwróci odpowiedź 250, może zawrzeć w nim pełną nazwę użytkownika i musi zawrzeć skrzynkę użytkownika. Musi więc być to jedno z poniższych:

- User Name <local-part@domain>
- local-part@domain

Kiedy argument podany w komendzie VRFY składa się z kilku adresów, serwer może zgłosić niejasność w adresie jakos całości albo wskazać na jeden z podanych w argumencie adresów. Dla przykładu odpowiedzi serwera na komendę VRFY mogą przyjąć postać:

- 530 User ambiguous
- 553- Ambiguous; Possibilities are 553-Joe Smith <jsmith@foo.com> 553-Harry Smith <hsmith@foo.c
- 553-Ambiguous; Possibilities 553- <jsmith@foo.com> 553- <hsmith@foo.com> 553 <dweep@foo.com

W normalnych okolicznościach klient, który otrzyma odpowiedź 553 oczekuje, że zwrócone wyniki będzie mógł zaprezentować użytkownikowi czyli będą one dla niego zrozumiałe. W powyższym przykładzie możliwe jest to tylko w przypadku 2 i 3 (pierwszy nic konkretnego poza informacją o błędzie nie mówi).

Drugie polecenie EXPN przyjmuje jako argument napis (string), który identyfikuje listę maili, a odpowiedź poprawna serwera może zawierać pełne nazwy użytkowników i musi podawać ich skrzynki mailowe.

Na niektórych hostach występuje problem z rozróżnieniem pomiędzy listą mailową, a aliasem do pojedynczej skrzynki mailowej, ponieważ struktura danych przechowująca informacje na ten temat może przyjmować oba typy wpisów i jest na przykład możliwe, że lista mailowa zawiera tylko jeden adres skrzynki. Próba weryfikacji takiego adresu poprzez VRFY, może udać się jedynie gdy serwer SMTP umie obsłużyć wiadomość wysłaną na ten adres poprzez dostarczenie jej do każdej skrzynki z listy. Jeżeli nie jest on w stanie tego zrobić zwróci jeden z następujących błędów

550 That is a mailing list, not a user lub też 252 Unable to verify members of mailing list

W przypadku wielolinijkowej odpowiedzi serwera na polecenie EXPN dokładnie jedna skrzynka mailowa powinna znaleźć się w lini.

Przykładowe polecenie EXPN i odpowiedzi na nie:

C: EXPN Example-People S: 250-Jon Postel <Postel@isi.edu> S: 250-Fred Fonebone <Fonebone@phys

C: EXPN Executive-Washroom-List S: 550 Access Denied to You.

Serwerowi nie wolno w odpowiedzi na VRFY lub EXPN zwracać odpowiedzi 250 dopóki nie dokona faktycznej weryfikacji adresu. Są jednak przypadki, w których adres wydaje się być prawidłowy, ale w czasie rzeczywistym nie może on

zostać zweryfikowany. Jest tak np. kiedy serwer służy jako pośredni dla innego serwera albo domeny. W takim przypadku serwer dokonuje jakby "pozornej weryfikacji". Sprawdza on poprawność składniową adresu, jak również może dokonać sprawdzenia domeny czy jest ona w zakresie domen do których maile może przekazywać. Odpowiedź serwera powinna więc uwzględnić, że jest to tylko częściowa weryfikacja adresu i zamiast statusu 250 powinien pojawić się status 252.

Implementacje serwera SMTP powinny zawierać przedstawione powyżej komendy. W celach bezpieczeństwa implementacje mogą dostarczać narzędzi (np. poprzez plik konfiguracyjny) do wyłączenia tych poleceń. Polecenia te były opcjonalne w RFC 821, ale obecnie powinny być wylistowane jako dodatkowe serwisy w odpowiedzi na EHLO.

3.2.5 Koniec sesji i połączenia

Jak już wcześniej zostało wspomniane sesja w protokole SMTP kończy się wraz z przesłaniem przez klienta polecenia QUIT. Serwer odpowiada pozytywnym kodem, po którym zamyka połączenie.

Serwerowi nie wolno zamknąć połączenia intuicyjnie poza 2 przypadkami:

- otrzyma polecenie QUIT i odpowie kodem 221
- po wykryciu potrzeby zamknięcia serwisu SMTP i przesłaniu kodu 421. Odpowiedź ta może być umieszczona po każdej komendzie lub asynchronicznie (przy założeniu, że klient odbierze ją zanim kolejne polecenie zostanie przesłane)

Klient SMTP, który spotka się z zamknięciem połączenia w wyniku okoliczności, których nie może obsłużyć, powinien traktować przeprowadzaną transakcję jak gdyby otrzymał od serwera odpowiedź 451 "Requested action aborted: error in processing".

3.2.6 Listy mailowe (mailingowe) i aliasy

Serwer SMTP powinien wspierać zarówno aliasy, jak i listy mailowe, które służą w celach dostarczenia wiadomości do większej ilości adresów.

Alias to inaczej pseudo-skrzynka, której nazwa przy odbieraniu maila tłumaczona jest na rzeczywisty adres (bądź adresy). Zmieniane jest tylko pole (RCPT TO), reszta wiadomości ("koperta" i ciało wiadomości) pozostaje nietknięta.

Lista mailowa zaś służy dostarczoniu albo przekazaniu maila do wszysktich adresów znajdujących się na liście. W wyniku tej operacji adres zwrotny na "kopercie" (MAIL FROM) musi być zmieniony na adres osoby albo innej jednostki, która sprawuje opiekę nad listą. Nagłówki wiadomości (w szczególności pole From) pozostają nietknięte.

3.2.7 Informace o trasie

Kiedy serwer SMTP otrzyma wiadomość do dostarczenia albo dalszego procesowania pozostawia w niej swój ślad. Ślad ten umieszczany jest na początku ciała wiadomości (nagłówki "time stamp" albo "Received").

Linia ta powinna się składać z następujących informacji:

- pola FROM, które powinno zawierać zarówno nazwe hosta źródłowego (tak ja w zaprezentowanym w ramach odpowiedzi na komende EHLO) i adres IP źródła, zdeterminowane z połączenia TCP
- pola ID, które może zawierać zgodnie z sugestiami w RFC 822 znaczke "@", ale nie jest to wymagane
- pola FOR, które może zawierać listę wpisów <path> kiedy w poleceniu RCPT zostało podanych ich kilka

Wcześniej dodane linijka Received nie powinna być zmieniana. Serwery SMTP powinny mieć przygotowaną linie Received do "wstrzyknięcia" do wiadomości. Nie wolno im zmieniać obecnych lini ani też wpisywać swojej lini w innym niż przeznaczone miejscu.

Przykładowa linia wygląda następująco:

```
Received: from mizar.astronet.pl ([127.0.0.1]) by localhost (mizar [127.0.0.1]) (amavisd-new, port 10024
```

W miarę rozwoju Internetu linie Received są bardzo istotne w celach wykrywania wystąpienia jakichkolwiek problemów.

Kiedy wiadomość jest ostatecznie dostarczana, serwer SMTP wstawia nagłówek "Return-path" na początku maila. Zachowanie to jest wymagane. Systemy mailowe SMTP muszą to implementować. "Return-path" to zachowane informacje z argumentu <reverse-path> z polecenia MAIL. Po tym oznaczeniu uznawane jest, że wiadomość opuściła środowisko SMTP, jednakże nie oznacza to obecnie, że została ona ostatecznie dostarczona. Może być transmitowana przez inny system mailowy.

3.2.8 Rozmiary i opóźnienia

W protokole SMTP występują obiekty, które mają wymagane minimalne bądź też maksymalne rozmiary. Każda implementacja musi być w stanie odbierać o takich wielkościach. Obiekty większe powinny być unikane. Klient może liczyć się z tym, że przy próbie transmisji zbyt dużych obiektów, może spotkać się z odmową ze strony serwera.

Obiekty o których mowa to:

- local-part - maksymalna długość nazwy użytkownika. Wynosie ona 64 znaki
- domain - maksymalna długość domeny. Wynosi to 255 znaków
- path - maksymalna długość argumentu reverse-path albo forward-path. Wynosi ona 256 znaków (wliczając w to interpunkcje i separatory elementów)
- comman line - maksymalna długość polecenia wraz z jego nazwą i znakiem <CRLF>. Wynosi ona 512 znaków. Rozszerzenia SMTP mogą posłużyć w celach zwiększenia tego limitu
- reply line - maksymalna długość lini odpowiedzi. Wynosia oan 512 znaków wliczając w to kod odpowiedzi oraz znak <CRLF>

- text line - maksymalna długość lini tekstu wliczając w to znak <CRLF>. Wynosi ona 1000 znaków. Może być ona rozszerzona przez dodatkowe serwisy SMTP.
- message content - długość zawartości wiadomości (wliczając w to nagłówki wiadomości jak i jego ciało). Musi mieć ona co najmniej 64K octets (czyli ile?). Odkąd wprowadzono standard Multipurpose Internet Mail Extensions (MIME) wielkość wiadomości wzrosła diametralnie, dlatego też zaleca się obecnie aby serwery SMTP nie nakłady żadnych ograniczeń na wielkość wiadomości
- recipients buffer - rozmiar buffora adresowego. Minimalna całkowita liczba odbiorców, która musi być zbuforowana to 100. Klient, który chce dostarczyć wiadomość do więcej niż 100 odbiorców powinien być przygotowany na transmisję maila w kawałkach po 100 odbiorców (jeżeli oczywiści serwer akceptuje wiadomości z taką ilością odbiorców)

Błędy związane z powyższymi ograniczeniami mogą być zgłoszone poprzez standardowe kody błędów. Przykłady takich błędów:

- 500 Line too long
- 501 Path too long
- 452 Too many recipients
- 552 Too much mail data

W ramach protokołu SMTP nie możemy także zapomnieć o opóźnieniach, które należy wziąć pod uwagę przy implementacji klienta. Klient musi repektować opóźnienia związane z poleceniami. Powinny one być w prosty sposób w kliencie konfigurowalne (w celach ewentualnych zmian i próby dostosowanie opóźnień do warunków panujących w sieci). Proponowane opóźnienia w kliencie:

- Początkowa wiadomość 220 od serwera: 5 minut
  Klient SMTP powinien mieć możliwość rozróżnienia pomiędzy problemami związanymi z TCP (zerwane połączenie) a opóźnieniami w otrzymaniu powitalnego 220. Wiele serwerów SMTP akceptuje połączenia, ale opoźnia ich obsługę dopóki load na maszynie nie spadnie do odpowiedniego poziomu
- Polecenie MAIL: 5 minut
- Polecenie RCPT: 5 minut
  Większe opóźnienie jest wymagane w przypadkach, gdy procesowanie obsługi listy mailowej czy aliasu nie następuje po akceptacji maila
- Inicjalizacja polecenie DATA: 2 minuty
  Potrzeba na czekanie na odpowiedź serwera "354 Start Input"
- Blok danych: 3 minuty
  Ewentualna potrzeba na oczekiwanie na zakończenie wywołanai TCP SEND do transmisji kolejnych porcji danych

- Zakończenie polecenia DATA: 10 minut
  Potrzebne w celach oczekiwania na odpowiedź 250 OK. Kiedy odbiorca otrzyma ostatni znak oznaczający koniec wiadomości, przeważnie zaczyna on procesowanie dostarczenia wiadomości do skrzynku użytkownika. Nieodpowiednie opóźnienie tutaj może skutkować dostarczeniem paru tego samego maila, ponieważ klient założy, że transakcja się nie udała (minie odpowiedni czas), a tak naprawde odbiorca zaakceptuje wiadomość, a jeszcze nie zdąży powiadomić o tym klienta.

W przypadku opoźnień serwera możemy mówić o stałym opóźnieniu 5 minut w oczekiwaniu na komendy od klienta.

3.2.9 Strategie wysyłania i odbierania

Przeważnie struktura implementacji hosta SMTP zawiera skrzynki użytkowników, jedną lub więcej kolejek wiadomości oraz jednego lub więcej demonów do odbierania i wysyłania maili. Dokładna struktura różni się w zależności od potrzeby użytkowników oraz liczby i wielkości list mailowych obsługiwanych przez host. Poniżej strategie, które okazały się bardzo pomocne w przypadku obsługi dużego ruchu mailowego:

- Wysyłanie:
  Głównym modelem dla klienta SMTP jest jeden lub kilka procesów, które cyklicznie próbują transmitować maile wychodzące. W typowym systemie, program, który komponuje wiadomość posiada odpowiednie metody, aby zwrócić natychmiastową uwagę na nowy element w wiadomościach wychodzących. Maile, które nie są transmitowane od razu są kolejkowane i cyklicznie wysyłane. Kolejka mailowa zawiera oprócz samej wiadomości "kopertę" czyli dodatkowe informacje służące do wysyłania.
  Nadawca po nieudanym wysłaniu powinien odczekać pewną jednostkę czasu zanim ponowi próbę transmisji maila. Generalnie jednostka ta powinna wynosić około 30 minut.
  Ponawianie wysłania następuje dopóki zakończy się ono sukcesem bądź też klient się podda. Przyjmuje się, że po 4-5 dniach nieudanych próbach, wiadomość zostaje uznana za niedoręczoną.
  Klient powinien trzymać listę hostów do których nie może dotrzeć i zapamiętywać opóźnienia do nich, zamiast nieustannie kolejkować maile przeznaczone do tych hostów. W ten sposób szybciej odrzuci maile, które nie mają szansy osiągnąć celu.
  Klient SMTP może zmniejszyć oczekiwanie w kolejce we współpracy z serwerem SMTP. Dla przykładu, jeżeli mail z danego adresu (domeny), został odebrany prawdopodobne jest, że maile kolejkowane dla tego adresu (domeny), mogą zostać wysłane.
  Klient SMTP może mieć kolejkę wielu wiadomości dla każdego nieosiągalnego hosta docelowego. Jeżeli każda z tych wiadomości w cyklu ponawiania będzie wysyłana system zostanie zablokowany na długi okres czasu, ponieważ klient może stwierdzić, że dostarczenie się nie udało jedynie po określonym czasie (do kilku minut). Nawet zmniejszenie opóźnienia do 1 minuty przy setkach wiadomości powoduje zamrożenie systemu.
  W przypadku wysyłania wiadomości do wielu odbiorców z tego samego

hosta klient powinien stosować taktykę MAIL, RCPT, RCPT, ... RCPT, DATA zamiast MAIL, RCPT, DATA, ..., MAIL, RCPT, DATA.

- Strategie odbioru
  SMTP server powinien oczekiwać nieustannie nadchodzących połączeń. Wymaga to oczywiście od implementacji obsługi wielu połączeń jednocześnie. Jak już wspomniano wcześniej obsługa procesu dostarczenia maila do konkretnej skrzynki może odbywać się nie tuż po obsłudze polecenia DATA, ale w odzielnym procesie. Zmniejszony jest wtedy czas oczekiwania klienta na ostateczne potwierdzenie zakończenia transakcji maila.

3.2.10 Rozwiązywanie adresu i przekazywanie maila

Zanim klient SMTP zidentyfikuje dokąd ma być posłany mail, musi nastąpić odpytanie serwera DNS o adres domeny. Oczekuje się, że podane w adresie nazwy domen są w pełni akceptowanymi nazwami domen (fully qualified domain name, FQDN). Pierwsze odpytanie serwera to zapytanie o rekorkdy MX związane z nazwą.

Rekord MX (Mail exchanger) to jeden z typów rekordów w serwerze domen DNS, który specyfikuje w jakis sposób maile powinny być routowane (przekazywane w sieci) przy użyciu protokolu SMTP. Każdy rekord MX ma następującą postać: [name] [ttl] IN MX preference host gdzie:

- name - nazwa komputera lub domeny czyli cel ostateczny naszego maila
- ttl - czas życia rekordu
- preference - komputer lub domena może mieć więcej niż jeden komputer wskazany jako odbiorca poczty, wpisana w tym polu liczba określa preferencję - im mniejsza jej wartość tym priorytet wyższy
- host - nazwa serwera pocztowego

Jeżeli przed rekordem MX zostanie znaleziony rekord CNAME, host który znajduje się pod tym rekordem jest następnie traktowany jak host, który był pierwotnie podany do poszukiwań w DNS-ie. Rekord CNAME to swego rodzaju alias w DNS-ie.

Jeżeli żaden rekord MX nie zostanie odnaleziony, ale za to odnaleziony zostanie rekord A, jest on traktowany jakby był rekordem MX z preferencją 0 (czyli najwyższym priorytetem).

Jeżeli uda się znaleźć jeden lub więcej rekordów MX dla danej nazwy, implementacjom systemów SMTP nie wolno używać żadnych rekordów A związanych z nazwą.

Kiedy odpytanie DNS-ów zostanie zakończone sukcesem, możemy w wyniku mapowania otrzymać kilka adresów, ze względu na możliwość kilku rekordów MX czy też multihoming (albo to i to). Ażeby zapewnić niezawodną transmisję maila, klient SMTP musi próbować (i ew. ponawiać próby) dla każdego z adresów wg porządku, aż uda się dostarczyć wiadomość. Klient może posiadać konfigurowalną liczbę maksymalnych alternatywnych adresów. Liczba ta powinna wynosić przynajmniej 2 adresy.

Jak wyżej wspomniano klient posyła maile do kilku adresów na podstawie uporządkowanej listy, która to zależna jest od priorytetów w rekordach MX oraz

od hostów z wieloma adresami. Im niższy priorytet tym serwer jest bardziej preferowany, zaś przy tych samych priorytetach dokonywany jest wybór losowy.

Adres hosta docelowego może okazać się nie pojedynczym adresem, ale listą adresów (multihomed host). Zadaniem serwera DNS jest zwrócenie już uporządkowanej listy adresów IP, a klient po kolei próbuje wysłać pod nie maile.

Obsługa alternatywnych adresów związanych z multihomingiem może zostać w kliencie ograniczona do określonej liczby albo w ogóle wyłączona.

3.3 Konstrukcja wiadomości

Główne informacje na temat wyglądu wiadomości są obecnie zebrane w kilku dokumentach. Jednym z głównych opisujących wygląd wiadomości bez rozszerzeń jest obecnie RFC 2822 (następca już uznanego za przedawniony RFC 822). Rozszerzenia wiadomości, które będą opisane później, zdefiniowane są w kolejnych dokumentach (seria dokumentów o MIME RFC2045, RFC2046, RFC2047, RFC2048, RFC2049).

Sama wiadomość, z punktu widzenia mocno niskopoziomowego, to po prostu seria znaków. Wiadomość zgodna ze standardem składa się ze znaków od 1 do 127 interpretowanych jako znaki kodowania US-ASCII. Wynika to oczywiście ze względów historycznych, gdyż, jak to juz było wspomniane, to w Stanach zostały posłane pierwsze wiadomości i to na tamte potrzeby stworzono pierwsze standardy.

Wiadomości są podzielone na linie. Linia to seria znaków, która jest ograniczona przez 2 znaki powrót karetki (CR - carriege return - w ASCII znak numer 13) i znak nowej linii (LF - line-feed - w ASCII znak numer 10). Znaki te z reguły występują razem w ramach wiadomości i oznaczane są jako CRLF.

Ogólnie wiadomość można podzielić na 2 części - pola z nagłówkami wiadomości (nazywane po prostu nagłówkiem wiadomości) oraz ciała wiadomości, które jest opcjonalne.

3.3.1 Ograniczenia w długości linii

Ilość znaków w linii jest ograniczona dwoma limitami. Maksymalna ilość znaków w linii (z pominięciem CRLF), musi być mniejsza niż 998 znaków, i nie powinna być większa niż 78 znaków.

Limit 998 znaków wynika z ograniczeń wielu obecnych implementacji, które zajmują się odbiorem, wysyłką oraz przechowywaniem wiadomości. Nie obsługują one po prostu dłuższych linii.

Bardziej konserwatywne ograniczenie związane z liczbą 78 znaków wynika z próby dostosowania wiadomości do wielu implementacji interfejsu użytkownika, które wyświetlają te wiadomości. Implementacje te mogą uciąć linie zawierające więcej niż 78 znaków.

3.3.2 Nagłówki wiadomości

Pole nagłówka składa się z:

- nazwy nagłówka po której następuje dwukropek ":"
- ciała nagłówka po które następuje znak CRLF

Nazwa nagłówka musi składać się z drukowalnych znaków US-ASCII (czyli tych, których kody są pomiędzy 33, a 126 włącznie) bez przecinka ",". Ciało nagłówka może zawierać wszystkie znaki oprócz CR i LF, chyba, że przy konstrukcji nagłówka użyto składania (ang. folding) opisanego niżej.

Same nagłówki można podzielić na posiadające (structered) lub nie posiadające dodatkowej struktury (unstructered).

Te drugie (unstructered) oprócz wcześniej nałożonych restrykcji co do występowania określonych znaków, nie mają dodatkowych obostrzeń. Semantycznie nie poddaje się ich dodatkowemu procesowaniu (chyba, że nagłówek należy poskładać [ang. unfold])

Niektóre nagłówki wymagają w ramach swojego ciała dodatkowej semantyki (struktury). Jeżeli dane ciało nagłówka nie jest zgodne z tą semantyką to taki nagłówek, a przez to także wiadomość, nie jest zgodna ze specyfikacją opisywaną w RFC 2822.

Jak już było wcześniej zaznaczone pojedynczy nagłówek to pojedyncza linia składająca się z nazwy, dwukropka i ciała nagłówka. Dla wygody, a także aby poradzić sobie z ograniczeniem na maksymalna długość linii, pojedynczy nagłówek może być rozłożony na kilka linii. Dzielenie to określamy angielskim słowem "folding". Dzielenie to polega na tym, że ciało nagłówka może zostać, w przypadku długiej lini, przeniesione do następnej. Linia kończona jest normalnym znakiem CRLF, ale nowa linia musi być rozpoczęta znakiem WSP (white space czyli spacją space [SP] - kod 32 lub tabulatorem horizontal-tab [HTAB] - kod 11). Przykładowo nagłówek:

Subject: This is a test

może być reprezentowany jako

Subject: This
 is a test

Dzielenie nagłówka na wiele lini może nastąpić w wielu miejscach jednak zaleca się wstawianie znaku CRLF pomiędzy tokenami, które znajdują się na najwyższym stopniu w semantyce nagłówka. Dla przykładu kiedy ciało nagłówka składa się z podzielonych przecinkami wartości, zaleca się łamanie lini po przecinku rozdzielającym struktury.

Jak już wcześniej było zaznaczone składanie nagłówka z wielu lini określane jest angieslkim słowem "unfolding".

3.3.3 Opis niektórych nagłówków

Niektóre z nagłówków wiadomości (jak to juz wcześniej wspomniano) posiadają pewną dodatkową strukturę i są przeznaczone do określonych celi. Dokładny opis semantyki ciała "zaawansowanych" nagłówków umieszczony jest w RFC 2822 i nie będzie tutaj poruszany, jednak warto wspomnieć jakie informacje niosą te nagłówki.

Date - nagłówek ten specyfikuje datę i czas, którą to twórca wiadomości wskazuje, że jest ona skończona i gotowa do wejścia w system dostarczania

poczty (czyli gotowa do wysłania). Dla przykładu może być to czas kiedy użytkownik tworząc maila klika przycisk "wyślij"

From, Sender, Reply-to - nagłówki określane są nagłówkami autora wiadomości. Pola te wskazują na skrzynkę (skrzynki) źródła wiadomości. W szczególności pole From określa autora (bądź autorów) wiadomości czyli skrzynkę (bądź skrzynki) osoby (lub osób) odpowiedzialnych za stworzenie maila. Nagłówek Sender określa skrzynkę odpowiedzialną za aktualną transmisję maila. Jeżeli twórcą wiadomości jest pojedyncza skrzynka oraz pole autora i transmitującego maila są identyczne nagłówek Sender nie powinien wystąpić. Inaczej oba (From i Sender) powinny się pojawić.
Reply-to (jeśli jest obecne) wskazuje skrzynkę (skrzynki), do których autor chciałby, aby słane były odpowiedzi. Jeżeli brakuje nagłówka Reply-to wtedy odpowiedzi powinny być słane na adres spod nagłówka From. Nagłówek From nie powinien zawierać żadnych adresów skrzynek nie należących do autora wiadomości.

To, Cc, Bcc - zwane są polami docelowymi. Określają one odbiorców wiadomości. Pole To określa adres głównego odbiorcy (bądź adresy głównych odbiorców). Pole Cc (nazwa pochodzi od Carbon Copy w rozumieniu tworzenia kopii przez pisarza przy użyciu kalki) zawiera inne adresy osób, które powinny otrzymać maila, chociaż to nie oni są głównymi adresatami. Bcc (Blind Carbon Copy) zawiera adresy odbiorców, które nie mają być pokazane innym odbiorcom. Są trzy sposoby użycia Bcc:

- Pierwszy z nich polega na usunięciu z wiadomości linijki Bcc (tuż przed wysłaniem wiadomości) i przesłanie jej kopii także do tych adresów z Bcc.
- Drugi sposób polega na posłaniu do adresatów z To i Cc wiadomości z usuniętą linijką Bcc, zaś adresy spod lini Bcc dostają maila w całości czyli z linijką Bcc. Tutaj pojawiają się różne zachowania, jeżeli w polu Bcc mamy kilka adresów. Niektóre implementacje modyfikują tak Bcc, że zawiera on za każdym razem tylko tego odbiorce do którego trafia mail.
- Ostatni sposób podobny jest do pierwszego tylko zamiast usuwania lini Bcc jest ona słana bez żadnego adresu

Sposób użycia Bcc zależy od implementacji klienta.

Message-ID, In-Reply-To, Regerences - chociaż opcjonalne pola te powinny być (i z reguły są) obecne w wiadomości. Message-ID zapewnia wiadomości unikalny identyfikator, który odnosi się do danej wersji danej wiadomości. Unikalność gwarantowana jest przez host, który generuje to id. Pola In-Reply-To i References są wykorzystywane przy tworzeniu odpowiedzi na wiadomość. Zawierają one identyfikator wiadomości oryginalnej (In-Reply-To) oraz identyfikatory innych wiadomości (References), gdy na przykład wiadomość jest odpowiedzią na odpowiedź. Tak więc In-Reply-To (jeśli istnieje) wskazuje nam na jaką wiadomość jest odpowiedzią ta wiadomość, zaś References zawiera wskazanie na "wątek" w którym jest wiadomość. W trakcie konstrukcji wiadomości nagłówki In-Reply-To i References tworzone są w następujący sposób:

In-Reply-To przepisuje zawartość z Message-ID. Jeśli istnieje kilka wiadomości nadrzędnych (parent message) wtedy In-Reply-To będzie zawierało wszystkie pola Message-ID poprzedników. Jeśli pole Message-ID nie istnieje In-Reply-To oczywiście nie jest tworzone.
References przejmuje wartość z wszystkich References (jeśli jakieś istnieją) po których przepisywane są wartości z Message-ID wszystkich wiadomości nadrzędnych. Jeśli wiadomości nadrzędne nie posiadają pola References, a posiadają pole In-Reply-To to miejsce References nadrzędnych wiadomości przejmuje In-Reply-To. Jeśli nadrzędne wiadomości nie mają żadnego z trzech pól o których tutaj mowa, nowa wiadomość odpowiedź nie będzie miała pola References.
Część implementacji parsuje pole References, aby odtworzyć przebieg dyskusji i utworzyć wątek z uporządkowanymi wiadomościami. Tak robi na przykład bardzo ostatnimi czasy popularny webowy klient poczty Gmail. Czytanie tak ułożonych wiadomości jest niesamowicie wygodne.

Subject, Comments, Keywords - pola te przeznaczone do celów informacyjnych powinny mieć formę czytelną dla ludzi. Subject to tytuł wiadomości. Comments zawiera bardziej szczegółowy opis wiadomości, zaś Keywords to porozdzielane przecinkami najważniejsze wyrazy opisujące wiadomość.

Return-Path, Received - pola te zawierają informacje na temat drogi wiadomości w systemie (tworzenie ich było opisane wcześniej)

Dodatkowo użytkownik może sobie sam zdefiniować i używać dodatkowych nagłówków. Oczywiście nazwy tych nagłówkó nie mogą się pokrywać z nazwami już używanymi. Dlatego przyjmuje się, że nagłówki dodatkowo definiowane (nie objęte specyfikacjami RFC 822 oraz RFC 2822) powinny być poprzedzane wyrażeniem "X-" jak na przykład:

X-Mailer: Microsoft Windows Mail 6.0.6001.18000

3.3.4 Ciało wiadomości

Ciało wiadomości to po prostu linie złożone ze znaków US-ASCII. Są tylko dwa wymogi co do znaków w ciele wiadomości:

- znaki CR i LF muszą występować razem w postaci CRLF
- długość lini musi być nie większa niż 998 znaków i powinna być nie większa niż 78 znaków (wyłączając CRLF)

3.3.5 Rozszerzenie wiadomości, MIME

Jak zostało wspomniane wyżej RFC 822 oraz RFC 2822 definiują w jaki sposób ma wyglądać ciało wiadomości w bardzo zawężony sposób. Nie uwzględniają one problemu związanego z obsługą wielu języków (tylko znaki US-ASCII są możliwe w ramach ciała wiadomości), nie wspominając już o możliwości przesyłania plików audio, wideo czy zdjęć. Dlatego też w 1996 powstało pierwsze RFC (2045) wychodzące na przeciw tym wszystkim problemom. Specyfikacja tych rozszerzeń znalazła się w aż 4 RFC (2045-2048), które to zostały nazwane Multipurpose Internet Mail Extensions w skrócie MIME.

MIME definiuje kilka dodatkowych nagłówków, które służą do opisu ciała wiadomości. Nagłówki te pojawiają się w 2 miejscach:

- jako normalne nagłówki na początku wiadomości
- jako nagłówki w ramach części wiadomości, o których mowa będzie później

Pierwszy z dodatkowych nagłówków to nagłówek MIME-VERSION, który obowiązkowo musi być zawarty w wiadomości (jeśli ma ona być zgodne ze standardem MIME) jako główny nagłówek, zaś nie jest konieczny w przypadku nagłówków w części wiadomości. Przykładowo nagłówek ten może wyglądać tak:

MIME-Version: 1.0

Kolejnym nagłówkiem w ramach wiadomości typu MIME jest nagłówek Content-Type, który definiuje w sposób zapisania danych w wiadomości. Argument występujący w nagłówku Content-Type jest nazywany "media type" (w wolnym tłumaczeniu rodzaj/typ danych). Przykładowo typ "image/xyz" mówi nam, że mamy do czynienia z danymi, które przedstawiają zdjęcie, nawet jeśli klient pocztowy nie jest w stanie rozpoznać typu xyz. Dodatkowe parametry przy typie danych nie mają wpływu na ich naturę, a jedynie dodatkowo określają te dane. Niektóre implementacje mogą nie rozpoznawać pewnych dodatkowych argumentów MIME i gdy taka sytuacja ma miejsce, argumenty te powinny być pomijane. Przykładowym dodatkowym parametrem dla typu "text", może być "charset" określający kodowanie tekstu w wiadomości.

Głównymi typy MIME (zdefiniowanymi w RFC 2046) to:

- text - wiadomość tekstowa. Podtyp "plain" określa wiadomość bez dodatkowego formatowania. Nie wymaga ona żadnego dodatkowego oprogramowania poza ewentualnie obsługą odpowiedniego kodowania. Możliwe oczywiście są inne podtypy chociażby html, który wskazuje, że dana wiadomość może być parsowana jakby była stroną html
- image - dane ze zdjęciem. Wymagają one dodatkowego oprogramowania w celu wyświetlenia zdjęcia. Podtypami są tutaj różne formaty zdjęć jak np. jpeg, gif czy png
- audio - dane dźwiękowe. Potrzebne jest dodatkowe oprogramowanie w celu odsłuchania dźwięku. Początkowyn podtypem jest "basic"
- video - dane video. Potrzebne jest dodatkowoe oprogramowanie w celu otwarcia pliku. Początkowym podtypem jest mpeg.
- application - dane binarne bliżej nie sprecyzowane, które mogą być poprawnie zinterpretowane przez dodatkowe oprogramowanie. Podtyp "octet-stream" powinien być używany jeżeli chcemy aby najprostszą akcją, która zostanie nam zaoferowana przez klienta pocztowego to zapis pliku na dysk. "Application" powinno być także używane np. w przypadku arkuszy kalkulacyjnych, materiałów PostScript czy innych formatów, które w prosty sposób nie są czytelne.
- multipart - typ ten definiuje, że wiadomość będzie złożona z kilku części. Typ ten posiada 4 podtypy:

  - mixed - obecnie bardzo często wykorzystywany w przypadku kiedy wiadomość składa się z kilku elementów różnych od siebie np. tekstu oraz zdjęcia.
  - alternative - służy do zdefiniowania tej samej treści w różnym formatowaniu np. tekst w postaci plain oraz html.
  - parallel - służy do wiadomości, której części mają być zaprezentowane jednocześnie
  - digest - wykorzystywany przy wiadomościach złożonych z części, z których każda ma inny typ

- message - enkapsulacja wiadomości. Zawartość takiej wiadomości jest sama w sobie wiadomością. Przykładowy podtyp "rfc822" definiuje wiadomość zgodną z RFC 822, zaś "partial" służy do zdefiniowania części wiadomości zgodnej z RFC 822 w celach transmisji jej fragmentów ze względu na jej dużą wielkości uniemożliwiającą przekazanie w jednym kawałku.

Wiadomości MIME, które nie posiadają nagłówka Content-type traktowane są jako wiadomości z nagłówkiem "Content-type: text/plain; charset=us-ascii".

Bardzo istotnym nagłówkiem jest nagłówek Content-Transfer-Encoding, który definiuje algorytm wg którego zakodowana została wiadomość (lub jej część). Transformacje zapisane w ramach nagłówka to znane algorytmy kodowania i nie zależą nigdy one od zewnętrznych informacji (spoza wiadomości). Jedną z ważniejszych funkcji kodowania wiadomości jest możliwość przesłania oryginalnie 8-bitowych danych za pomocą 7-bitowych znaków US-ASCII.

Dostępne formy kodowania wiadomości to:

- 7bit - dane reprezentowane w postaci krótkich lini ze znakami US-ASCII
- 8bit - dane reprezentowane w postaci krótkich linni ale mogą występować znaki spoza US-ASCII
- binary - oprócz możliwości występowania znaków spoza US-ASCII może brakować podziału na krótkie linie
- quoted-printable - szczegółowy opis podany później
- base64 - szczegółowy opis podany później
- ietf-token - forma kodowania wyspecyfikowana w określonym RFC i zarejestrowna w IANA (Internet Assigned Numbers Authority) (wcześniej IETF - Internet Engineering Task Force)
- x-token - programiści mogą użyć własnego kodowania w nagłówku Content-Transfer-Encoding np. Content-Transfer-Encoding: x-my-new-encoding. Dodatkowe formy kodowania mogą być wyspecyfikowana w RFC. Wymogi takiej specyfikacji znajdują się w RFC 2048

Prawidłowy nagłówek Content-Transfer-Encoding musi zostać użyty. Nadanie wiadomości z 8-bitowymi danymi jako 7bit jest niedozwolone, zaś niekodowane dane muszą być "oflagowane" jako binary. Transfer niekodowanych danych protokołem SMTP jest obecnie uważany jako niezgodny ze specyfikacją.

W odróżnieniu od typów i podtypów MIME tworzenie własnych nazw kodowań dla nagłówka Content-Transfer-Encoding jest mocno odradzane.

Nagłówek Content-Transfer-Encoding tyczy się całego ciała wiadomości. Jeżeli zaś znajdzie się w części wiadomości to dotyczy tylko jej. Jeżeli wiadomość lub jej część jest typu multipart, wtedy Content-Transfer-Encoding może przyjmować tylko 7bit, 8bit lub binary.

Content-Transfer Encoding jest bezpośrednio powiązany z drugim nagłówkiem jakim jest Content-Type, chociaż wydawałoby się, że są to całkowicie rozdzielne rzeczy. Po pierwsze niektóre kodowania mogą być stosowne w przypadku niektórych rodzajów typów MIME, jak choćby w przypadku 8bit-owego transportu (jeżeli serwer SMTP udostępnia taką możliwość) nie wymagany jest jakiekolwiek kodowanie dla tekstu o określonym kodowaniu znaków, podczas gdy dla 7bit-owego transportu tekstu takie kodowanie jest już niezbędne gdyż mogą wystąpić w nim znaki spoza US-ASCII. Po drugie niektóre typy MIME wymagają różnych typów kodowania w zależności od warunków. Przykładowo wiele dokumentów PostScript posiada krótkie linie z 7bit-owymi danymi i dlatego też nie wymagają one dodatkowego kodowania. Inne dokumenty PostScriptowe (szczególnie te używające mechanizmu kodowania binarnego 2 poziomu) mogą być przekazane tylko przy pomocy kodowania binary. I wreszcie po trzecie, ścisła specyfikacja pomiędzy typami MIME, a kodowaniem skutecznie łączy specyfikację protokołu aplikacji z specyfikacją niskopoziomowego transportu. Nie jest to jednak porządane, gdyż zmuszałoby deweloperów typów MIME do posiadania znajomości rodzajów transportu i ich ograniczeń.

Jeżeli jakaś wiadomość lub jej część posiada nierozpoznawalny Content-Transfer-Encoding musi być traktowana jakby wartość nagłówka Content-Type wynosiła "application/octet-stream" niezależnie od tego jaką wartość rzeczywiście posiada ten nagłówek.

Kodowanie Quoted-Printable pozostawia wszystkie bajty o wartości mniejszej od 128, które nie są znakami sterującymi ASCII (z wyjątkiem znaku równości =) bez zmian, a pozostałe (z wyjątkiem znaku tabulacji poziomej - HT, kod 9) zamienia na napis zakodowany w ASCII reprezentujący kod szesnastkowy danego bajtu poprzedzony znakiem równości =. Sam znak równości w celu uniknięcia wieloznaczności jest zastępowany ciągiem =3D. Dodatkowe reguły rządzą kodowaniem końców linii, m.in. "miękkim" łamaniem linii, oraz reprezentacją linii kończących się białymi znakami.

Base64 służy do kodowania ciągu bajtów przy pomocy ciągu znaków. Kodowanie to przypisuje 64 wybranym znakom (patrz tabelka niżej) wartości od 0 do 63. Ciąg bajtów poddawany kodowaniu dzielony jest na grupy po 3 bajty. Ponieważ bajt ma 8 bitów, grupa 3 bajtów składa się z 24 bitów. Każdą taką grupę dzieli się następnie na 4 jednostki 6-bitowe. Istnieją więc dokładnie 64 możliw wartości każdej takiej jednostki. Wszystkim tym jednostkom są przypisywane znaki na podstawie podanego niżej arbitralnie ustalonego przypisania.

Wartość	Znak	Wartość	Znak	Wartość	Znak
0	A	22	W	44	s
1	B	23	X	45	t
2	C	24	Y	46	u
3	D	25	Z	47	v
4	E	26	a	48	w
5	F	27	b	49	x
6	G	28	c	50	y
7	H	29	d	51	z
8	I	30	e	52	0
9	J	31	f	53	1
10	K	32	g	54	2
11	L	33	h	55	3
12	M	34	i	55	3
13	N	35	j	56	4
14	O	36	k	57	5
15	P	37	l	58	6
16	Q	38	m	59	7
17	R	39	n	60	8
18	S	40	o	61	9
19	T	41	p	62	+
20	U	42	q	63	/
21	V	43	r	pad	=

Jeśli rozmiar wejściowego ciągu bajtów nie jest wielokrotnością liczby 3, to stosowane jest dopełnianie (na końcu wynikowego ciągu dodawana jest taka ilość symboli dopełnienia (pad), aby ten miał długość podzielną przez 4).

Widać, że dane zakodowane przy pomocy base64 na maszynie, która używa 8-bitowego słowa do reprezentacji znaków powiększają swój rozmiar o 33

Jeśli rozmiar wejściowego ciągu bajtów nie jest wielokrotnością liczby 3, to stosowane jest dopełnianie (na końcu wynikowego ciągu dodawana jest taka ilość symboli dopełnienia (pad), aby ten miał długość podzielną przez 4).

Widać, że dane zakodowane przy pomocy base64 na maszynie, która używa 8-bitowego słowa do reprezentacji znaków powiększają swój rozmiar o 33

Wspomnieć też należy o nagłówku Content-ID, który jest w zastosowaniu podobny do Message-ID i pozwala na referencję pomiędzy różnymi wiadomościami. Tak samo jak Message-ID wartość zawarta w Content-ID musi pozostać unikalna.

Ostatnim dodatkowym nagłówkiem jaki zawarty jest w standardzie MIME to nagłówek Content-Description. Wartość tego nagłówka to tekst US-ASCII. Opisuje on po prostu ciało danego załącznika np. "rysunek Galaktyki Andromedy".

Dodatkowe nagłówki opisujące załączniki mogą zostać w przyszłości zdefiniowane. Nazwa każdego nowego nagłówka powinna rozpoczynać się od "Content-" np. "Content-Newone".

Jak było wcześniej wspomniane typ MIME multipart służy do tworzenia wiadomości składających się z załączników różnego rodzaju. Minimalnie może ona zawierać jeden załącznik. Każdy z nich poprzedzony jest linią podziału (boundary line), zaś po ostatnim z załączników występuje linia kończąca. Po każdej lini rozpoczynającej załącznik znajdują się nagłówki określające go (opi-

sane wcześniej), następnie występuje pusta linia, a za nią znajduje się ciało. Ze względu na posiadanie nagłówków i ciała załącznik podobny jest do wiadomości zgodnej z RFC 822, jednak może on na przykład nie zawierać nagłówków. Taki załącznik traktowany jest (podobnie jak wiadomość MIME bez nagłówka Content-Type) jako posiadający nagłówek Content-Type typu text/plain; charset=US-ASCII. Linia dzieląca załączniki wiadomości nie może pojawić się w tekście ciała załącznika. Dlatego też jest niezwykle istotne, aby oprogramowanie klienckie tworzyło wiadomości zawierające bardzo unikalne linie podziału. Przykładowa linia podziału może wyglądać następująco:

- - - -=_NextPart_001_0064_01C82582.9D9EB7E0

Poniżej przykładowa wiadomość (ze względu na czytelność wiadomości niektóre nagłówki zostały usunięte, a załączniki zostały zmniejszone). Składa się ona z 2 załączników:

- multipart/mixed z boundary
  - - - -=_NextPart_000_0063_01C82582.9D9C6DF0), który sam w sobie jest typu multipart/alternative (text/plain i text/html) z boundary
  - - - -=_NextPart_001_0064_01C82582.9D9EB7E0
- image/jpeg - zdjęcie w formacie jpeg

Treść wiadomości:

Date: Mon, 12 Nov 2007 23:20:30 +0100
From: "jerzy_bielecki«jbielecki@klc.vectranet.pl>
To: <redakcja@astronet.pl>
MIME-Version: 1.0
Message-ID: <006701c8257a3d2100b045e7584e@compurbae2b8cf>
Subject: Luna
Content-Type: multipart/mixed;
boundary=- - - -=_NextPart_000_0063_01C82582.9D9C6DF0"

This message is in MIME format. Since your mail reader does not understand this format, some or all of this message may not be legible.

- - - - - -=_NextPart_000_0063_01C82582.9D9C6DF0

Content-Type: multipart/alternative;
boundary=- - - -=_NextPart_001_0064_01C82582.9D9EB7E0"

This message is in MIME format. Since your mail reader does not understand this format, some or all of this message may not be legible.

- - - - - -=_NextPart_001_0064_01C82582.9D9EB7E0

Content-Type: text/plain;
charset="iso-8859-2"
Content-Transfer-Encoding: quoted-printable

Witam!

mentu fotografii) !
Powi=EAkszy=B3em dodatkowo jasn=B1 "plamk=EA"i uczytelni=B3em filtrem grafi=
cznym - interesuj=B1cy efekt ...
Pomo=BFecie ?

Pozdrawiam - Jerzy W. Bielecki

_ _ _ _ _ _ _ _ _ _ _ _
Poczta sprawdzona przez G DATA AntiVirus
Wersja: AVK 18.584 z 12.11.2007
Informacje: www.gdata.pl

- - - - - -=_NextPart_001_0064_01C82582.9D9EB7E0
Content-Type: text/html;
charset="iso-8859-2"
Content-Transfer-Encoding: quoted-printable

<!DOCTYPE HTML PUBLIC -//W3C//DTD HTML 4.0 Transitional//EN»
<HTML><HEAD>
<META http-equiv=3DContent-Type content=3D"text/html; charset=3Diso-8859-2»
<META content=3D"MSHTML 6.00.6000.16544ńame=3DGENERATOR>
<STYLE></STYLE>
</HEAD>
<BODY bgColor=3D#ffffff>
Witam!
Powi=EAkszy=B3em dodatkowo jasn=B1 "plamk=EA"i uczytelni=B3em filtrem grafi=
cznym - interesuj=B1cy efekt ...
Pomo=BFecie ?

Pozdrawiam - Jerzy W. Bielecki
_ _ _ _ _ _ _ _ _ _ _ _
Poczta sprawdzona przez G DATA AntiVirus
Wersja: AVK 18.584 z 12.11.2007
Informacje: www.gdata.pl

</BODY></HTML>

- - - - - -=_NextPart_001_0064_01C82582.9D9EB7E0- -

- - - - - -=_NextPart_000_0063_01C82582.9D9C6DF0
Content-Type: image/jpeg;
name="IMG_7661xLunaPlusAJW9plusResizeLunaPlusUczytelnienia.jpg"

Content-Transfer-Encoding: base64
Content-Disposition: attachment;
filename="IMG_7661xLunaPlusAJW9plusResizeLunaPlusUczytelnienia.jpg"

/9j/4AAQSkZJRgABAQEASABIAAD/4RAXRXSSTaS
hpZgAASUkqAAgAAAAJAA8BAgAGAAAAegAASasS
AAAAgAAAABIBAwABAAAAAQAAABoBBQABAAAS
AgAUAAAAsAAAABMCAwABAAAAAQAAAGmHBA

- - - - - -=_NextPart_000_0063_01C82582.9D9C6DF0- -

3.4 Obecne wykorzystanie protokołu i jego forma

3.4.1 SMTP-AUTH

Protokół SMTP w obecnej formie wykorzystuje wiele rozszerzeń, których powstanie było naturalnym wynikiem rozwoju Internetu. Podstawowym brakiem protokołu SMTP w formie z RFC 2821 jest brak jakiejkolwiek autoryzacji czyli potwierdzenia tożsamości klienta podłączającego się do serwera. Klient, który poprawnie przejdzie proces autoryzacji może użyć serwera, do którego jest podłączony, w celach przekazania wiadomości (ang. relay). Serwer po przyjęciu tej wiadomości jest odpowiedzialny za jej dostarczenie. Autoryzacja zapobiega m.in. przed relay-owaniem poczty od nieautoryzowanych użytkowników czyli tzw. spammerów. Przykładowa sesja SMTP z poprawną autoryzacją:

S: 220 smtp.example.com ESMTP server ready C: EHLO jgm.example.com S: 250-smtp.example.com S: 250 AUTH CRAM-MD5 DIGEST-MD5 C: AUTH FOOBAR S: 504 Unrecognized authentication type. C: AUTH CRAM-MD5 S: 334 PENCeUxFREJoU0NnbmhNWitOMjNGNndAZWx3b29kLmlubm9zb2Z0LmNvbT4= C: ZnJlZCA5ZTk1YWVlMDljNDBhZjJiODRhMGMyYjNiYmFlNzg2ZQ== S: 235 Authentication successful.

Widać więc, że w całej komunikacji pojawia się dodatkowe polecenie AUTH. Serwer najpierw informuje klienta, że autoryzacja jest obsługiwana. Następnie klient korzysta z polecenia AUTH, którego składnia jest następująca:

AUTH SPACE auth_type [SPACE (base64 / "=")] *(CRLF [base64]) CRLF

i próbuje dowiedzieć się jaka forma autoryzacji jest dostępna. Serwer odpowiada kodem 334 jeżeli dany typ autoryzacji obsługuje lub 504 w przeciwnym wypadku. W przypadku poprawnej odpowiedzi wysyła on zakodowany base64 ciąg znaków, na podstawie którego klient przesyła inny ciąg znaków także zakodowany base64. Ciąg ten z reguły zawiera hasło i nazwę użytkownika połączone z danymi odkodowanymi z serwera. W przypadku prawidłowego ciągu znaków od klienta serwer dokonuje autoryzacji (odpowiedź 235).

Mimo rozszerzenia SMTP-AUTH cały czas protokół SMTP nie daje gwarancji, że wiadomość pochodzi od osoby, za którą podaje się nadawce. Wystarczy przecież złamać hasło (w przypadku słabych haseł) metodą brute-force i podszywać się pod daną osobę.

3.4.2 Phishing i obrona przed nieautoryzowanymi mailami

Coraz częstszym problemem jest też phishing (zwany też spoofingiem). Polega on na konstruowaniu wiadomości, w których adres nadawcy (podany w nagłówku) oraz jej inne części są tak spreparowane jakby wydawały się mailem z innego źródła. Najczęściej zmienianymi nagłówkami są From, Return-Path, Reply-To. Maile takie najczęsciej powiązane są z tzw. phishingiem webowym. Polega to na spreparowaniu wiadomości i zawarciu w niej odniesienia do strony internetowej, która przypomina nam znaną strone internetową, którą jest również odpowiednio spreparowana. Często wiadomości te próbują podszywać się pod wiadomości przesyłane nam z naszych banków, a linki w nich zawarte prowadzą do stron przypominających strony logowań do systemu bankowego. Po zalogowaniu się na takiej stronie internata staje się ofiarą, ponieważ jego login i hasło przekazywane jest osobom, które spreparowały maila, jak i stronę internetową.

Aby zapobiec takim sytuacjom powstały rozszerzenia, których celem jest potwierdzenie tożsamości nadawcy wiadomości. Podobną funckję spełniają na przykład certyfikaty na szyfrowanych stronach internetowych. Niektóre z tych rozszerzeń to:

- SPF - Sender Policy Framework - uwierzytelnia tylko adres MAIL FROM. Sprawdza, czy e-mail przychodzący z danego serwera pocztowego może stosować określoną nazwę domenową w MAIL FROM. Opiera się na infrastrukturze DNS oraz rekordach TXT

- DKIM - DomainKeys Identified Mail - uwierzytelnia treść i niektóre nagłówki wiadomości.
  Opiera się na cyfrowym podpisywaniu treści wiadomości i części nagłówków (podpis w oddzielnym nagłówku) oraz infrastrukturze DNS do dystrybucji kluczy publicznych. Powstał z połączenia Yahoo! DomainKeys i Cisco Identified Internet Mail.

- CSV - Certified Server Validation - uwierzytelnia tylko parametr polecenia HELO/EHLO.
  Określa poziom zaufania przy połączeniu nadchodzącym z innego MTA na podstawie jego adresu IP oraz parametru polecenia HELO lub EHLO. Stosuje infrastrukturę DNS oraz rekordy SRV.

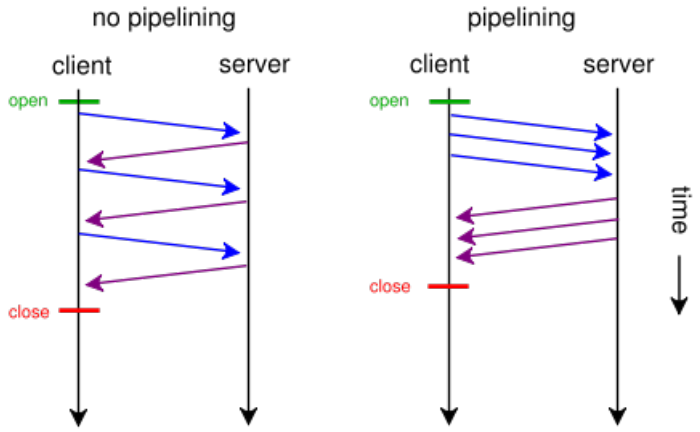
- Microsoft Sender ID - uwierzytelnia MAIL FROM oraz From.
  Syntaktycznie zgodny z SPF. Rozszerza użycie SPF do nagłówków From:. Oparty na SPF oraz propozycji Microsoftu z 2004 r.: Caller ID for E-Mail.

W obecnej formie każda z powyższych propozycji (prócz mało efektywnego i praktycznie niestosowanego CSV) wprowadza poważne ograniczenia w używaniu poczty.

Większość wymusza stosowanie wybranych rozwiązań na administratorach serwerów, które kontaktują się z chronionym serwerem. Żadna z nich nie jest natomiast standardem, a implementacja zmian jest często kłopotliwa (np. nie każdy MTA umożliwia stosowanie SRS). Skuteczność powyższych rozwiązań jest zbyt niska w porównaniu z ograniczeniami, jakie wprowadzają. Redukują spam i ataki wirusów w stopniu mniejszym, niż inne rozwiązania które nie mają tak poważnych efektów ubocznych (np. greylisting).

3.4.3 Pipelining

Ważnym rozszerzeniem protokołu SMTP jest opisany w RFC 1854 tzw. pipelining. Pipeling jest cechą komunikacji w ramach protokołu polegającą na wykonaniu kilku żądań na raz, bez czekania na poszczególne odpowiedzi z serwera.



Serwer SMTP informuje klienta, że obsługuje rozszerzenie pipeliningu poprzez zawarcie słowa kluczowego PIPELINING w odpowiedzi na EHLO klienta. Klient, jeżeli potrafi obsługiwać to rozszerzenie, może trasmitować grupy komend bez oczekiwania na rezultat każdej z nich. Polecenia RSET, MAIL FROM, SEND FROM, SOML FROM, SAML FROM i RCPT TO mogą występować dowolnie w grupie, zaś EHLO, DATA, VRFY, EXPN, TURN, QUIT oraz NOOP występują jako ostatnie, ponieważ powodzenie tych poleceń lub błąd w nich powodują zmianę stanu, na którą reaguje klient. Dodatkowe polecenia wprowadzone przez rozszerzenia protokołu także występują jako ostatnie, chyba że ich specyfikacje określają dodatkowe możliwości.

Klient, który implementuje pipelining musi sprawdzić wszystkie statusy odpowiedzi z serwera SMTP. Przykładowo jeżeli odpowiedzią serwera na RCPT TO będzie brak akceptacji dla adresu, klient musi sprawdzić status odpowiedzi DATA. Nie może on zakładać, że polecenie DATA nie zostanie zaakceptowane w wyniku braku akceptacji polecenia RCPT TO.

Przykładowa sesja z użyciem pipeliningu ma następującą postać:

S: <wait for open connection>
C: <open connection to server>
S: 220 innosoft.com SMTP service ready
C: EHLO dbc.mtview.ca.us
S: 250-innosoft.com
S: 250 PIPELINING
C: MAIL FROM:<mrose@dbc.mtview.ca.us>
C: RCPT TO:<ned@innosoft.com>
C: RCPT TO:<dan@innosoft.com>
C: RCPT TO:<kvc@innosoft.com>
C: DATA
S: 250 sender <mrose@dbc.mtview.ca.us> OK
S: 250 recipient <ned@innosoft.com> OK

S: 250 recipient <dan@innosoft.com> OK
S: 250 recipient <kvc@innosoft.com> OK
S: 354 enter mail, end with line containing only "."
...
C: .
C: QUIT
S: 250 message sent
S: 221 goodbye

3.4.4 Szyfrowanie

Komuniakcja pomiędzy klientem, a serwerem w protokole SMTP odbywa się za pomocą komend przesyłanych otwartym tekstem. W wielu przypadkach wiadomość transportowana jest przez wiele routerów i serwerów, co do których możemy nie mieć zaufania. Może nastąpić przechwycenie informacji, albo jej zmiana.

Próba rozwiązania tego problemu doprowadziła do powstania mechanizmu TLS, bardziej znany jako SSL, który rozszerza komunikację TCP o ochronę prywatności i autentykację.

W celu obsłużenia TLS-u stworzono rozszerzenie do protokołu SMTP i wprowadzono dodatkowe polecenie STARTTLS. Odpowiedzi klienta na tę komendę mogą być następujące:

- 220 Ready to start TLS
- 501 Syntax Error (no parameters allowed) w przypadku gdy do bezparametrowego polecenia STARTTLS klint podał parametr
- 454 TLS not available due to temporary reason

W przypadku otrzymania odpowiedzi 220 klient powinien zacząć negocjacje TLS przed wszystkimi innymi poleceniami. Po negocjacji TLS (szczegóły działania rozszerzenia TLS w wersji 1.0 zwarte jest w RFC 2246, obecnie rozwijana jest wersja 1.1 opisana w RFC 4346) obie strony uzgadniają czy kontynuować dalszą komunikację. W przypadku niepowodzenia autentykacji lub szyfrowania i tak może nastąpić do dalszej komunikacji klient-serwer i akceptacji przez serwer wiadomości. Jeżeli serwer stwierdzi, że wynegocjowany poziom zaufania jest zbyt niski powinien przesłać klientowi komendę SMTP QUIT.

Przykładowa sesja z uwzględnieniem negocjacji TLS może przyjmować następującą postać:

S: <waits for connection on TCP port 25>
C: <opens connection>
S: 220 mail.imc.org SMTP service ready
C: EHLO mail.ietf.org
S: 250-mail.imc.org offers a warm hug of welcome
S: 250 STARTTLS
C: STARTTLS
S: 220 Go ahead
C: <starts TLS negotiation>

C & S: <negotiate a TLS session>
C & S: <check result of negotiation>
C: <continues by sending an SMTP command>
. . .

Należy jednak zaznaczyć, że jeżeli komunikacja pomiędzy klientem, a serwer była przeprowadzona przy użyciu TLS-u, nie oznacza to, że cały transport wiadomości był szyfrowany. Po drodze niestety może dojść do nieszyfrowanego lub nieautoryzowanego przekazywania wiadomości.

4 Dzisiejsze narzędzia do filtracji protokołu SMTP

4.1 Konieczność wprowadzenia filtracji

Coś o spamie ilości spamu procentowo konieczności zarządzaniem mailami w firmach itd. itp.

4.2 Produkty komercyjne

Obecnie na rynku znajduje się sporo produktów służących do filtracji w ramach protokołu SMTP. Niektóre z nich są częścią większych projektów, inne zostały zaprojektowane tylko w tym celu.

4.2.1 Clearswift - MIME sweeper for SMTP

MIMEsweeper for SMTP e-mail software to bardzo wyrafinowany system do zarządzania pocztą email. Według twórców systemu żaden inny system nie oferuje tak obszernych możliwości kompleksowego zarządzania infrastrukturą maili. Dzięki rozwojowi przez ponad 20 lat systemów do zarządzania e-mailami, obecny produkt firmy Clearswift stał się idealnym rozwiązaniem dla firm, które pragną w pełni ochronić komunikację e-mail.

Firma Clearswift zajmuje się sektorem zabezpieczeń komunikacyji i dostarcza filtry emailowe i webowe dla około 25 milionów użytkowników skupionych wokół 17 tysięcy firm. Została założona w roku 1982, a jej obecna siedziba mieści się w Theale, niedaleko Reading w Wielkiej Brytani.

Dwie najważniejsze funkcjonalności jakie dostarcza MIMEsweeper for SMTP to:

- Routowanie i relayowanie wiadomości w ramach SMTP w zależności od zdefiniowanych zasad
- Procesowanie wiadomości w ramach domeny w oparciu o zdefiniowane zasady bezpieczeństwa

W ramach polityki bezpieczeństwa firmy MIMEsweeper umożliwia skonfigurowanie wielu zasad bezpieczeństwa, które zostały podzielone na 3 główne typy:

1. Polityka deploymentu - określa ona sposób implementacji MIMEsweepera w danej sieci. Zależna jest ona od:
   - architektury sieci w ramach określonej domeny
   - sposobu podłączenia sieci do Internetu
   - ilości osób zarządzających polityką bezpieczeństwa i samym systemem
   - elastyczności systemu
2. Polityka routowania i relayowania poczty. Można tutaj określić:
   - z jakich hostów będziemy akceptować maile
   - jakie hosty mogą łączyć się do naszego serwera i używać go do relayowania poczty

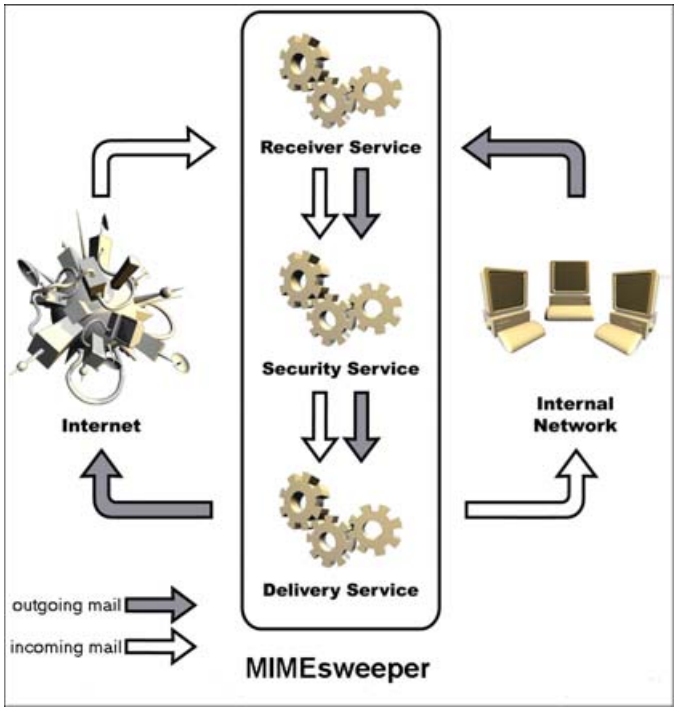
- z jakich hostów albo adresów poczty nie przyjmujemy
- dozwoloną ilość odbiorców
- dozwoloną wielkość wiadomości email
- ilość wiadomości email jakie MIMEsweeper może jednocześnie przetwarzać

3. Polityka ochrony zawartości. Określa ona reguły przetwarzania przez system maili takie jak:

   - kto może wysyłać i odbierać maile
   - dokąd maile mogą być wysyłana albo skąd odbierane
   - jaki rodzaj wiadomości wykryć
   - jaki rodzaj tekstu i jakie obiekty analizować w wykrytych wiadomościach
   - jaki rodzaj akcji podjąć po analizie
   - co zgłosić po wykryciu maila
   - jakie informacje o systemie śledzić
   - jakie przetwarzane wiadomości śledzić
   - jaką analizę zawartości i jej wykrycie śledzić

Całość oprogramowania MIMEsweeper podzielona jest na serwisy, które umożliwiają realizację wcześniej wymienionych zadań. Serwisy te to:

- serwis odbierający - zgodnie z ustalonymi zasadami polityki bezpieczeństwa waliduje przychodzące połączenie, odbiera maila i przekazuje do serwisu zabezpieczeń do dalszego procesowania
- serwis zabezpieczeń - sprawdza zawartość maila i konfrontuje go z skonfigurowaną wcześniej polityką bezpieczeństwa
- serwis dostarczający - przekazuje pocztę do odbiorców
- serwis odpowiedzialny za infrastrukturę - zajmuje się monitorowaniem systemu, konfiguracją i kontrolą zadań nie objętych przez inne serwisy
- serwis analityczny - złożonych z dwóch odrębnych serwis. Pierwszy z nich zbiera dane o systemie i przekazuje je do drugiego serwisu odpowiedzialnego za stworzenie z nich bazy danych w odpowiedniej postaci
- system śledzenia - odpowiedzialny za zbieranie wszystkich danych, których wcześniej zostały zdefiniowane w systmie do śledzenia



4.2.2 Aladdin - eSafe Mail

eSafe firmy Aladdin to bardzo elastyczne i łatwe w zarządzaniu rozwiązanie dla mniejszych i większych systemów informatycznych. Pozwala na instalację w wielu konfiguracjach w zależności od potrzeb klienta. Sam produkt eSafe zawiera w sobie:

- eSafe Gateway - wprowadza dodatkowo warstwę bezpieczeństwa. W przykładowym modelu może ona być umieszczona za firewallem przed serwerem proxy. Chroni sieć przed atakami, wirusami, botami czy trojanami.
- eSafe Web - ochrona przed wszystkimi atakami związanymi z technologiami webowymi. Poza standardową ochroną przed wirusami, oprogramowaniem śledzącym użytkowników (spyware) czy filtrowaniem adresów URL, rozwiązanie to dostarcza chociażby blokowanie znanych exploitów, politykę bezpieczeństwa obiektów ActiveX czy blokowanie stron HTML zawierającymi zdefiniowane słowa kluczowe
- eSafe Web SSL - analiza szyfrowanych zawartości webowych czyli m.in. inspekcja szyfrowanego ruchu (HTTPS, SSL, TLS), automatyczne unieważnianie cerfytikatów na podstawie CRL (Certificate Revocation List) czy obsługa blacklist i whitelist dla administratorów.
- eSafe Mail - kompleksowa ochrona wiadomości email
- eSafe Modules - dodatkowe modułu rozszerzające oprogramowanie eSafe takie jak:

- AppliFilter - monitoruje i blokuje nieautoryzowany ruch internetowy aplikacji z grup takich jak P2P (peer-to-peer), IM (instant messaging) czy Spyware.
- URL Filtering - monitoruje i blokuje dostęp do zdefiniowanych jako nieodpowiednie stron internetowych
- Advanced Antispam - filtr niechcianej poczty. Wysyła na bieżąco zapytania o weryfikację reputacji nadawcy, klasyfikując na tej podstawie pocztę oraz nadawców i ewentualnie blokując przesłanie szkodliwej zawartości do adresata. Uzyskiwana skuteczność osiągnęła w tym względzie poziom 98% przy czym badania nowego rozwiązania wykazują, że zakres błędu (np. zatrzymanie oczekiwanej poczty lub poczty nie będącej spamem) wynosi 0,05%.
- SpywareNeutralizer - skaner oprogramowania typu spyware
- Advanced Reporter - modul do tworzenia raportów na podstawie logów z eSafe

Produktem związanym z filtracją SMTP w ramach projektu eSafe, pomijając dodatkowy moduł do zaawansowanego zarządzania polityką antypsamową Advance Antispam, jest oczywiście eSafe Mail. Sercem tego produktu jest Dual Engine Email Security.

Ten dualny silnik pracuje w trybie czasu rzeczywistego - jego zadaniem jest błyskawiczna weryfikacja reputacji nadawcy wiadomości i poddanie analizie wzorców dystrybucji przychodzących maili. Rozwiązanie monitoruje ruch w światowym Internecie pod kątem wykrywania podejrzanych, masowych maili, badając przede wszystkim kluczowe punkty przepływu poczty elektronicznej, m.in. wśród dostawców Internetu (ISP). Dzięki temu rozwiązaniu eSafe może najszybciej jak to możliwe zidentyfikować spam oraz złośliwe oprogramowanie, zanim trafią one do firmowej sieci. Narzędzie monitoruje masowo wysyłane wiadomości pod kątem zagrożenia spamem, phishingiem, złośliwymi kodami, jednocześnie bez blokowania np. także masowo wysyłanych i subskrybowanych newsletterów. W połączeniu z dualnym silnikiem antyspamowaym wspomnianym wcześniej (Advance Antispam), rozwiązanie eSafe staje się potężnym narzędziem do ochrony poczty email.

4.2.3 Surfcontrol - E-mail Filter

Pakietu E-mail Filter firmy Surfcontrol to system kompleksowego zarządzania pocztą elektroniczną. Dostępny jest jako samodzielna brama SMTP lub jako rozszerzenie do serwera poczty Exchange firmy Microsoft. Oprogramowanie to umożliwia:

- opanowanie spamu
- ochronę danych poufnych
- filtrowanie obraźliwych treści
- dodawanie kolejnych warstw ochronnych przed wirusami
- optymalizację zasobów serwera pocztowego

Oprogramowanie E-mail Filter umożliwia w ramach administracji polityką bezpieczeństwa m.in.:

- tworzenie reguł tej polityki za pomocą kreatora reguł metodą drag-and-drop
- wyodrębnianie, usuwanie, opóźnianie lub wysyłanie do adresata dowolnej przesyłki, która narusza wcześniej zdefiniowane reguły
- przeglądanie zawartości poczty przez administratorów
- wprowadzenie filtrowania sieci na poziomie przedsiębiorstwa z możliwością zdalnego zarządzania
- automatyczne powiadamianie wyznaczonej osoby o próbie naruszenia polityki
- generowanie raportów w zależności od potrzeb użytkownika
- filtrowanie poczty wewnętrznej (tylko w przypadku serwera Exchange 5.5) oraz wiadomości przychodzących/wychodzących

Filtrowanie treści wiadomości wykorzystuje jeden z najobszerniejszych w branży słownik w celach sugerowania potencjalnie obraźliwych, istotnych dla firmy, zastrzeżonych lub innych treści wysokiego ryzyka.

Firma Surfcontrol udostępnia dodatkowe moduły do oprogramowania E-Mail Filter. Są nimi:

- Anti-Spam Agent (poprzednio: RiskFilter) - chroni firmę przed spamem
- Virtual Learning Agent - chroni przed przypadkowym lub zamierzonym wysłaniem poczty zawierającej zastrzeżone dane
- Virtual Image Agent - w oparciu o wprowadzone kryteria, filtruje niewłaściwe obrazki zawierające treści dla dorosłych
- Anti-Virus Agent - skaner antywirusowy, który ochroni organizację przed plagą wirusów w przesyłkach pocztowych

4.3 Produkty open-source

Oprócz produktów komercyjnych na rynku znajdują się też produkty darmowe.

4.3.1 Amavisd-new i dodatkowe filtry

Amavisd-new to wysokiej jakości interfejs pomiędzy MTA, a filtrami zawartości: skanerem wirusów i filtrem spamu. Oprogramowanie napisane jest w języku Perl. Komunikuje się z MTA poprzez (E)SMTP lub LMTP (Local Mail Transfer Protocol) albo przy pomocy innego dodatkowego oprogramownia. Częstymi filtrami zawartości użytymi wraz z amavisd-new są chociażby SpammAssassin do walki ze spamem oraz skaner antywirusowy ClamAV. SpamAssassin to napisany głównie w perlu zestaw skryptów do skanowania zawartości poczty elektronicznej i oceny prawdopodobieństwa czy dana wiadomość jest spamem, czy też nie. W swoich oznaczeniach program stosuje metodę punktową, gdzie im wyższa

ocena, tym większe prawdopodobieństwo że treść wiadomości jest niepożądana. Posiada możliwość "uczenia się" wiadomości chcianych i nie chcianych co przy zastosowaniu odpowiedniej ilości przykładowych wiadomości pozwala na osiągnięcie całkiem zadowalających efektów filtracji. Clam AntiVirus jest zestawem narzędzi antywirusowych dla systemów UNIX, dostępnym na licencji GPL oraz zaprojektowanym z myślą o skanowaniu wiadomości e-mail na poziomie serwerów pocztowych. Pakiet dostarcza wielu narzędzi, takich jak elastyczny i skalowalny wielowątkowy demon, skaner z linii poleceń, i zaawansowane narzędzie do automatycznych aktualizacji baz sygnatur wirusów. Głównym elementem pakietu jest silnik antywirusowy, dostępny w formie współdzielonej biblioteki.

4.3.2 Smtp vilter

Smtp-vilter to filtr zawartości dla serwera smtp sendmail. Potrafi on wykryć wiadomości z wirusami oraz spam. Wykryte na podstawie reguł wiadomości mogą zostać przepuszczone, odrzucone lub zaznaczone. Filtr opiera się na dodatkowym oprogramowaniu do skanowania zawartości. Komunikacja z sendmail odbywa się poprzez milter API.

5 Opracowany filtr poczty SMTP

Przedstawione powyżej oprogramowanie komercyjne to bardzo duże systemy, w których filtrowanie wiadomości jest jednym z mniejszych programów lub modułów. Oprogramowanie to oczywiście nie jest za darmo i często zakupienie i wdrożenie takiego systemu wiąże się z niemałymi kosztami. Firmy, które nie mogą pozwolić sobie na takie rozwiązanie często wybierają wspomniane produkty open-sourcowe. Produkty te jednak nie są kompletne, często zależne są od dodatkowych programów. Rozwój ich nie jest tak dynamiczny, jak w przypadku produktów komercyjnych, i często spoczywa na głowie tylko kilku programistów. Dlatego powstała idea stworzenia własnego oprogramowania związanego z filtracją protokołu SMTP. Za język programowania służący do implementacji projektu została wybrany Java.

5.1 Założenia projektu

Głównym zadaniem oprogramowania jest udostępnienie możliwości segregacji wiadomości na podstawie zdefiniowanych reguł. Reguły pozwalają na określenie jakie elementy wiadomości mają być analizowane i jakie akcje podjąć po analizie. Komunikacja z oprogramowaniem odbywa się poprzez protokół SMTP - klient pragnący wysłać wiadomość do hosta X komunikuje się najpierw z filtrem i jemu przekazuje wiadomość. Wiadomość taka zostaje poddana analizie i ewentualnie odrzucona lub przesłana do hosta docelowego.

5.2 Moduły projektu

Filtr składa się z następujących modułów:

- odbierającego i wysyłającego
- parsera wiadomości
- parser reguł
- analizatora wiadomości
- kolejki

Ogólny schemat przepływu wiadomości w filtrze jest następujący:

1. Wiadomość jest odbierana przez moduł do odbierania wiadomości, a następnie trafia do kolejki przychodzącej
2. Wiadomości z kolejki wyjmowane są i przekazywane do analizatora. Wiadomości analizowane są współbieżnie, a ilość ich jednocześnie procesowanych zależy od konfiguracji filtra.
3. Analizator jest ścisle związany z parserem wiadomości oraz parserem reguł. Wyniki parsowania wiadomości i parsowania reguł są przekazywane do analizatora w postaci odpowiednio zdefiniowanych struktur. Na tej podstawie analizator decyduje jaką akcję podjąć. Możliwe są 3 scenariusze:
    - wiadomość zostanie zatrzymana przez filtr

- wiadomość zostanie przepuszczona i przekazana dalej bez żadnych zmian
- wiadomość zostanie przepuszczona, a jej struktura ulegnie zmianie - dodane zostaną odpowiednie nagłówki

4. W przypadku przepuszczenia wiadomości przekazywana jest ona do kolejnej kolejki wychodzącej
5. Wiadomości z kolejki wychodzącej obsługiwane są przez moduł wysyłający. Zajmuje się on wyjmowaniem wiadomości z kolejki i wysyłaniem ich do hosta docelowego. Wiadomości z tej kolejki obsługiwane są współbieżnie, a liczba równocześnie procesowanych wiadomości zależy od konfiguracji filtra. Gdy wiadomość zostanie poprawnie wysłana usuwana jest z kolejki. W przypadku nieudanego wysłania wiadomości jest ona także usuwana z kolejki - wielokrotne próby wysłania wiadomości nie są obsługiwane.

Rysunek jak to wygląda.

5.2.1 Parser wiadomości

Parser wiadomości to główna część projektu - najbardziej wymagająca programistycznie i najistotniejsza w sensie dalszej analizy wiadomości. Na jego potrzeby stworzone zostały następujące struktury (wyrażone klasami w języku Java):

- MimeMessage - główna klasa reprezentująca wiadomość email (typu MIME). Klasa ta posiada referencję na listę obiektów typu MimeMessageHeader reprezentującą nagłówki wiadomości oraz referencję do obiektu klasy Part
- Part - jest to abstrakcyjna klasa, z której dziedziczą MimePart oraz MimeMultiPart. Instancje tej klasy reprezentują ciało (część) wiadomości, które następuje po nagłówkach
- MimePart - klasa ta reprezentuje części wiadomości MIME prostego typu - przykładowo text/html. Instancja obiektu MimePart jest tworzona zarówno w przypadku, gdy podstawowy typ wiadomości jest prosty, jak i w przypadku prostego typu załączników. Przez podstawowy typ wiadomości rozumiany jest typ podany w głównych (czyli znajdujących się przed ciałem wiadomości) nagłówkach wiadomości
- MimeMultiPart - klasa ta reprezentuje załącznik wiadomości typu multipart. Klasa posiada jedynie referencje do obiektów typu MimePart oraz najbardziej charakterystyczną informację dla typu multipart jaką jest boundary. W klasie tej znajduje się główna metoda do parsowania wiadomości, która na podstawie boundary rekursywnie wyodrębnia załączniki z wiadomości
- MimeMessageHeader - klasa reprezentująca nagłówek wiadomości. Składa się z nazwy nagłówka oraz ciała nagłówka.
- MimeMessageHeaders - klasa reprezentująca listę nagłówków wiadomości. Oprócz referencji do nich zawiera główną metodę parsującą nagłówki z uwzględnieniem foldingu.

Oprócz tych najważniejszych klas reprezentujących odpowiednie elementy z wiadomości istotne było zaprojektowanie także klas pomocniczych. Klasy te to:

- Preamble - reprezentuje preambułę czyli tekst znajdującą się pomiędzy załącznikami nagłówkami wiadomości typu multipary, a nagłówkami wiadomości
- ContentType - klasa reprezentująca nagłówek content-type. Zawiera metodę, która odtwarza nagłówek do postaci prostej, jeżeli jego fragmenty są podzielone na części. Umożliwia także wyodrębnienie, jeżeli istnieje, boundary z nagłówka.

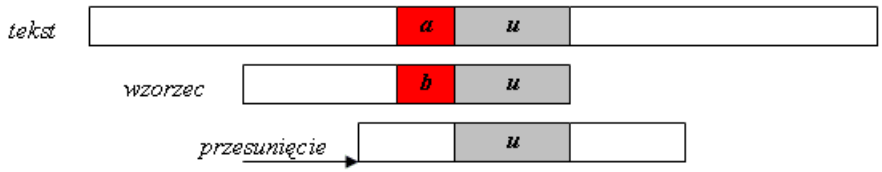
Główna metoda parsująca wiadomość i wyodrębniająca z wiadomości typu multipart załączniki opiera się na wyszukiwaniu wzorca w tekście. Wyszukiwanym wzorcem jest boundary, zaś tekstem ciało wiadomości. Algorytm wykorzystywanym przy tym wyszukiwaniu to algorytm **Boyer-Moore'a**.

Uważany jest on za najbardziej efektywny algorytm wyszukiwania wzorca. Wykorzystuje się go powszechnie w aplikacjach gdzie występuje polecenie "szukaj" albo "zamień".

Algorytm przyjmuje założenie, że między długością tekstu (n), a wzorcem (m), zachodzi zależność n>=m.

Skanowanie w tym algorytmie odbywa się od prawej strony wzorca. W przypadku niespasowania danego znaku bądź dopsaowania całego wzorca stosuje się dwie funkcje z wcześniej wyliczonymi wartościami. Są tą: "good-suffix shift" oraz "bad-character shift". łanie na przykładach. Załóżmy, że część wzorca i teskstu zaznaczona jako u pasuje do siebie. Natomiast na kolejnej pozycji występuje różnica. W zależności od sytuacji możemy przesunąć się w poszukiwaniu o różną wartość:

- "good-suffix shift" - jeżeli wzorzec zawiera w sobie kolejne wystąpienie u to przesuwamy się aby pokrywało się ono z tekstem

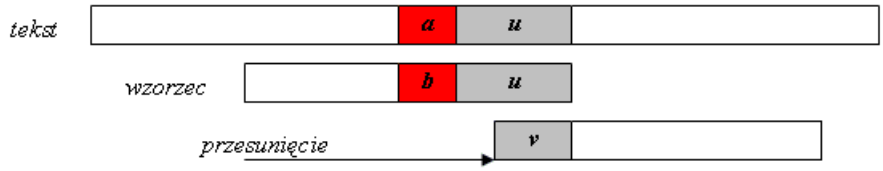


tekst: ccccccbbabbaababbacc
wzorzec: abbaabba

Niedopasowanie występuje na 4 znaku licząc od końca, ale wzorzec zawiera w sobie fragment juz dopasowany (bba), zatem przesuwamy się tak by kolejny taki fragment we wzrocu spasował się z tekstem.

tekst: ccccccbbabbaababbacc
wzorzec: abbaabba

- "good-suffix shift" - wzorzec nie zawiera w sobie kolejnego wystąpienia u to przesuwamy się tak by suffix wzorca spasował się maksymalnie z jego dotychczas spasowanym prefiksem



tekst: cccccbbabbaababbacc
wzorzec: baabba

Niedopasowanie występuje na 4 znaku licząc od końca, wzorzec nie zawiera w sobie fragmentu juz dopasowanego (bba), ale możemy dopasować jak najdłuży suffix (będzie to ba)

tekst: cccccbbabbaababbacc
wzorzec: baabba

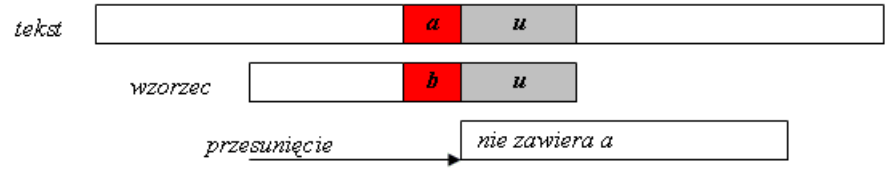
- "bad-character shift" - przesuwamy się tak by pierwszy znak od prawej w szukanym wzorcu spasował się z aktualnie rozpatrywanym znakiem w tekście

tekst: ccccccbbabbaababbacc
wzorzec: cbcaabba

Niedopasowanie występuje na 4 znaku licząc od końca (w tekście mamy "c" we wzorcu mamy "a"), zatem przesuwamy wzorzec w ten sposób by pierwsze "c" od prawej spasowało się z tym, które "popsuło" nam to dopasowanie

tekst: ccccccbbabbaababbacc
wzorzec: cbcaabba

- "bad-character shift" - przesuwamy się tak by pierwszy znak od prawej w szukanym wzorcu spasował się z aktualnie rozpatrywanym znakiem w tekście, jeżeli takiego znaku nie ma to ustawiamy koniec wzorca bezpośrednio za nim



tekst: ccccccebbabbaababbacc
wzorzec: cbcaabba

Niedopasowanie występuje na 4 znaku licząc od końca (w tekście mamy "e" we wzorcu mamy "a"), we wzorcu ęńie występuje zatem przesuwamy

się tak by wzorzec był bezpośrednio za nim

tekst: ccccccebbabbaababbacc
wzorzec: cbcaabba

W algorytmie stosujemu zawsze maksymalną wartość wynikającą z "good-suffix shift" oraz "bad-character shift".

5.2.2 Parser reguł

Parser reguł uruchamiany jest w trakcie ładowania konfiguracji. Klasa do tworzenia z pliku konfiguracyjnego reguł, na podstawie, których dokonywana jest później analiza, znajduje się w klasie Configuration. Posiada ona metodę loadConfiguration, która analizuje zawarty w poszczególnych liniach tekst i w zależności od niego:

- tworzy reguły analizy wiadomości
- konfiguruje parametry filtra

Możliwe do utworzenia reguły to:

- reguła definiująca konieczność wystąpienia lub zakaz występowania konkretnego nagłówka (lub nagłówków) w wiadomości - reprezentowana przez klasę MimeMessageHeaderPresenceRule
- reguła definiująca wielkość załącznika (lub załączników) w wiadomości - reprezentowana przez klasę MaxPartSizeRule
- reguła definiująca zakaz lub konieczność obecności w nagłówku zdefiniowanego słowa kluczowego - reprezentowana przez klasę KeywordSearchHeaderRule
- reguła definiująca akceptowane formaty plików - reprezentowana przez klasę AttachmentAnalizeRule
- reguła definiująca typy plików jakie będą poddawane głębszej analizie (przykładowo pliki zip mogą być sprawdzane pod kątem poprawności sumy kontrolnej) - reprezentowana przez klasę FileAnalizerRule
- reguła definiująca zakaz lub konieczność obecności w załączniku zdefiniowanego słowa kluczowego - reprezentowana przez klasę KeywordSearchAttachmentRule

Wszystkie przedstawione klasy dziedziczą z bazowej klasy Rule.
Przykładowa linia pliku konfiguracyjnego, z której mogą powstać dane reguły zostaną podane poźniej w ramach konfiguracji rozdziału o konfiguracji filtra.

5.2.3 Analizator wiadomości

Analizator wiadomości służy określeniu akcji jaka zostanie wykonane (odrzucenie wiadomości lub jej przepuszczenie) na podstawie reguł, które zostały stworzone w trakcie ładowania konfiguracji. Wiadomość dostarczana do analizatora zostaje oczywiście wcześniej sparsowana przez moduła parsujący wiadomości.

Posiada ona odpowiednią strukturę, która ułatwia jej analizę. Analizator jest więc elementem spajającym reguły, które są jego elementem nieodzownym elementem, oraz sparsowaną wiadomość. Analizator jest reprezentowany przez klasę Analizer.

5.2.4 Kolejka

Kolejka służy do przetrzymywania wiadomości odebranych przez filtr. Wiadomości przed analizą trafiają do kolejki wiadomości przychodzących. Następnie każda wiadomość wyjmowana z kolejki trafia do parsera i jest poddawana analizie. Po analizie wiadomości, jeżeli nie są one odrzucone, trafiają do kolejki wychodzącej i tam czekają na dalszą transmisję. Kolejka udostępnia API umożliwiające wyjmowanie i wkładanie do niej wiadomości. Reprezentowana jest ona przez klasę Queue.

5.2.5 Moduł odbierający i wysyłający

5.3 Kompilacja, konfiguracja i uruchomienie projektu

5.4 Testy wydajnościowe

6 Spostrzeżenia, wnioski

Literatura

[Cro82] David H. Crocker. *RFC822 - STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES*. 1982.