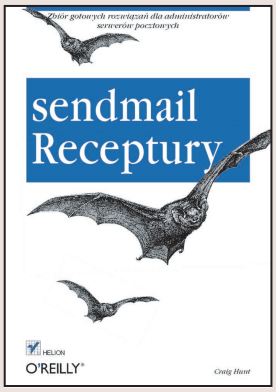
sendmail. Receptury



Autor: Craig Hunt
Tłumaczenie: Adam Jarczyk
ISBN: 83-7361-475-3
Tytuł oryginału: sendmail Cookbook
Format: B5, stron: 416

Mimo że sendmail jest najpowszechniej używanym uniksowym serwerem obsługującym pocztę elektroniczną, perspektywa jego konfigurowania wzbudza u administratorów sieci zdecydowanie nieprzyjemne uczucia. Języki wykorzystywane przy konfigurowania sendmaila są bardzo złożone i wykorzystywane stosunkowo rzadko – podczas instalacji i wstępnej konfiguracji modułu. Z tego właśnie powodu wielu administratorów nie ma zbyt wielu okazji do poznania mechanizmów konfiguracji sendmaila. Kiedy podczas pracy zachodzi nagła potrzeba zmiany konfiguracji tego programu, wszyscy odruchowo sięgają po dokumentację i spędzają wiele godzin na poszukiwaniu w niej rozwiązania swojego problemu.

Książka „sendmail. Receptury" to zbiór gotowych rozwiązań. Dzięki zawartym w niej poradom administrator szybko i sprawnie rozwiąże niemal każdy problem związany z konfiguracją sendmaila. Nie trzeba już przedzierać się przez setki stron dokumentacji. Koniec z metodą prób i błędów. Każda receptura, poza omówieniem problemu i przedstawieniem gotowego kodu, zawiera także analizę rozwiązania, która jest bardzo pomocna przy dostosowywaniu kodu do własnych potrzeb.

- Instalacja i wstępna konfiguracja sendmaila
- Doręczanie i przekazywanie dalej poczty
- Tworzenie list wysyłkowych
- Maskarada
- Kierowanie wiadomości
- Ochrona kont pocztowych przed spamem
- Uwierzytelnianie za pomocą protokołu AUTH
- Korzystanie z protokołu OpenSSL i obsługa certyfikatów
- Zarządzanie kolejką
- Zabezpieczenia sendmaila

„sendmail. Receptury" to obowiązkowa pozycja dla administratora sieci. Pozwala na zaoszczędzenie nie tylko pracy, ale i czasu.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

6

Kontrola nad spamem

6.0. Wstęp

Spam. Można go poznać na pierwszy rzut oka. Każdy spotkał się z tymi absurdalnymi reklamami systemów inwestowania, pornografii, specyfików ziołowych i wszystkiego innego, co tylko możemy sobie wyobrazić. Wielu ekspertów klasyfikuje jako spam wszystkie niezamawiane handlowe wiadomości e-mail (UCE — ang. *Unsolicited Commercial Email*) i istotnie takie wiadomości zaśmiecają skrzynki pocztowe na całej kuli ziemskiej. Jednakże legalni ogłoszeniodawcy, którzy oferują realne opcje wycofywania reklam, nie stanowią najgorszego problemu.[1] Prawdziwym problemem są spamerzy. Od prawowitych ogłoszeniodawców różni ich to, że spamerzy ukrywają swoją tożsamość i nadużywają cudzych systemów pocztowych. Można ich poznać po tym, że:

- Ukrywają źródło swoich wiadomości e-mail, aby nikt nie wiedział, kto je wysyła. Pełnoprawni ogłoszeniodawcy z dumą wyświetlają nazwę swojej firmy w adresie nadawcy. Każdy ogłoszeniodawca, który tego nie robi, ma przypuszczalnie powody, by ukrywać swoją tożsamość i swoje działania, i powinien być uważany za spamera.
- Nadużywają usług w cudzych systemach, wykorzystując je jako nieautoryzowane przekaźniki.
- Wysyłają informacje, których nigdy nie zamawialiśmy. Na listę wysyłkową spamera nie zapisujemy się osobiście, a gdy wiadomość od spamera zawiera opcję rezygnacji z przysyłania reklam, jej użycie nie powoduje usunięcia użytkownika z żadnej listy wysyłkowej. Zamiast tego opcja służy do gromadzenia adresów e-mail sprzedawanych następnie innym spamerom

Każdy z nas powinien pomagać w walce ze spamem, gwarantując, że nasz system pocztowy nie będzie nadużywany przez spamerów z zewnątrz, i że lokalni użytkownicy nie będą wysyłać niechcianych wiadomości.

---

[1] Puryści pomstują na opcje rezygnacji z reklam, lecz takie metody mogą być skuteczne, jeśli naprawdę są implementowane.

Spamerzy do ukrywania swojej tożsamości używają otwartych przekaźników poczty (ang. *open relay*). Receptury przedstawione w rozdziale 3. pokazują, jak można skonfigurować przekaźnik poczty. Po każdej zmianie konfiguracji powinniśmy przetestować przekaźnik tak jak w rozdziale 3., aby uniknąć nadużycia systemu.

Pierwszym krokiem w walce z lokalnymi spamerami jest stworzenie zasad dopuszczalnego użytkowania, które będą zakazywać rozsyłania spamu i zdefiniują działania, jakie podejmiemy w celu powstrzymania emisji spamu. Zasady powinny jasno informować, że poczta e-mail nie jest prywatna i może podlegać rejestrowaniu, skanowaniu i filtrowaniu. Ponieważ takie zasady są oficjalne, muszą zostać zaaprobowane i wydane przez kierownictwo oraz przejrzane i zaaprobowane przez biuro prawne. Zasady zawsze sprawiają kłopoty, ponieważ nie są rozwiązaniami technicznymi i wymagają udziału kierownictwa. Jednakże ten krok jest niezbędny, ponieważ daje nam uprawnienia do analizy poczty. Po wprowadzeniu zasad wszyscy użytkownicy muszą wyrazić na nie zgodę, aby otrzymać konto użytkownika.

Oprócz zapewnienia, że nasz system nie będzie czynnie przyczyniał się do produkcji spamu, istnieją narzędzia techniczne, które mogą posłużyć do aktywnego zwalczania spamu. sendmail zawiera szereg narzędzi do walki z niechcianymi reklamami:

- Baza danych *access* blokuje pocztę z i do określonych systemów.
- Listy „czarnych dziur" (ang. *blackhole lists*) DNS blokują wiadomości od przypuszczalnych spamerów i z otwartych przekaźników poczty.
- Programy filtrujące pocztę, np. procmail, są dostępne z sendmaila i potrafią filtrować pocztę na podstawie treści wiadomości
- Przetwarzanie nagłówka w sendmailu może wykrywać niekonsekwentne i źle zbudowane nagłówki.
- sendmail automatycznie przeprowadza testy poprawności logicznej, na przykład sprawdza, czy domena nadawcy może być rozwiązana przez DNS.

Baza danych access

Baza danych *access* pozwala na precyzyjną kontrolę nad przekazywaniem i doręczaniem poczty. Użycie funkcji *access_db* dodaje bazę danych *access* do konfiguracji sendmaila:

```
FEATURE(`access_db')
```

Domyślnie baza danych *access* jest typu hash i mieści się w pliku */etc/mail/access.db*. Za pomocą opcjonalnego pola argumentu w funkcji *access_db* możemy zmienić te domyślne ustawienia, jak w przykładzie:

```
FEATURE(`access_db', `btree -T<TMPF> /var/mail/access')
```

Ten wpis zmienia typ bazy danych, dodaje opcję `-T<TMPF>` związaną z obsługą tymczasowych problemów z wyszukiwaniem i zmienia ścieżkę do pliku bazy danych. Ustawienia domyślne powinniśmy zmieniać tylko w razie absolutnej konieczności.

Każdy wiersz w bazie danych *access* zawiera dwa pola: klucz i wartość zwracaną. Gdy używamy tej bazy danych do kontroli nad spamem, kluczem jest adres, a wartością zwracaną akcja, którą sendmail powinien podjąć w związku z wiadomością przychodzącą lub wychodzącą spod podanego adresu. Gdy używaliśmy bazy danych *access* w rozdziale 3., wartością zwracaną było, jak łatwo się domyślić, słowo kluczowe `RELAY`. Receptury znajdujące się w niniejszym rozdziale wykorzystują bazę danych do ustalenia, czy wiadomość powinna być zaakceptowana lub doręczona pod podany adres. Do tego celu potrzebny będzie zestaw innych wartości zwracanych. Słowa kluczowe stosujące się do kontroli spamu zostały wymienione w tabeli 6.1.

Tabela 6.1. Słowa kluczowe bazy danych access służące do kontroli nad spamem

Słowo kluczowe	Czynność
`OK`	Akceptuje wiadomości do i spod podanego adresu.
`DISCARD`	Odrzuca wszystkie wiadomości do i spod podanego adresu.
`REJECT`	Zwraca komunikat o błędzie i odrzuca wszystkie wiadomości do i spod podanego adresu.
`ERROR:`*`dsn:kod tekst`*	Odrzuca wiadomość z podanym kodem odpowiedzi i komunikatem o błędzie.
`HATER`	Dodaje zestawy reguł `check_mail` i `check_relay` dla wiadomości do podanego odbiorcy.
`FRIEND`	Pomija zestawy reguł `check_mail` i `check_relay` dla wiadomości do podanego odbiorcy.

Polecenie `OK` powoduje zaakceptowanie przez sendmail wiadomości ze źródła określonego przez pole klucza, niezależnie od innych warunków. Na przykład, jeśli nazwy hosta podanej w adresie nie można rozwiązać w DNS-ie, sendmail będzie akceptował wiadomość, nawet jeśli funkcja *accept_unresolvable_domains* nie będzie włączona. `OK` akceptuje wiadomości przeznaczone do doręczenia lokalnie, lecz nie nadaje przywileju przekazywania. Do tego celu byłoby wymagane słowo kluczowe `RELAY`, opisane w rozdziale 3.

Słowo kluczowe `REJECT` zwraca standardowy komunikat błędu do źródła i odrzuca wiadomość. Akcja `DISCARD` odrzuca wiadomość bez odsyłania komunikatu o błędzie do nadawcy. Wiele organizacji zajmujących się zwalczaniem spamu nie zgadza się z ideą odrzucania wiadomości „po cichu", ponieważ ich zdaniem nie zniechęca to spamerów. Z punktu widzenia spamera wiadomość sprawia wrażenie odebranej, więc dalej będzie nadawał swoje śmieci. Inne organizacje preferują odrzucanie wiadomości bez odpowiedzi, ponieważ uważają, że odpowiedź na wiadomość w dowolnej postaci jest dla spamera weryfikacją adresu i zachęca go do dalszych ataków. Oba podejścia chronią użytkowników przed spamerem, jednakże użycie akcji `REJECT`, która zwraca komunikat o błędzie do źródła wiadomości, pomaga w tych przypadkach, gdy pełnoprawna wiadomość zostanie błędnie sklasyfikowana jako spam, ponieważ informuje nadawcę, że wiadomość została odrzucona.

Akcja `REJECT` wysyła domyślny komunikat błędu. Możemy użyć słowa kluczowego `ERROR`, aby odrzucić wiadomość, odsyłając własny komunikat o błędzie:

```
example.com          ERROR:5.7.1:550 Nie przekazujemy spamu
```

W tym przypadku komunikat o błędzie zwracany do nadawcy będzie brzmiał „Nie przekazujemy spamu". Komunikat zawiera kod powiadomienia o stanie doręczenia 5.7.1 i kod błędu SMTP 550. Powinniśmy zastosować poprawny kod DSN z dokumentu RFC 1893, zgodny z kodem błędu z RFC 821 i komunikatem tekstowym.[2] Format komunikatu błędu przedstawiony w tabeli 6.1 to `ERROR:`*`dsn:kod tekst`*. Taki format jest zalecany, lecz nie jest wymagany. Możemy pominąć słowo kluczowe `ERROR` lub kod DSN, jednakże taki starszy format błędu jest odradzany. Aby zapewnić zgodność z przyszłymi wersjami sendmaila, powinniśmy używać słowa kluczowego `ERROR` i kodu DSN.

Słowa kluczowe `FRIEND` i `HATER` stosują się tylko wtedy, gdy funkcja *delay_checks* jest włączona do kontrolowania, kiedy mają być zastosowane zestawy reguł `check_mail` i `check_relay`. Słowo kluczowe `FRIEND` w wartości zwracanej pozwala wiadomościom, które normalnie zostałyby odrzucone przez `check_mail` lub `check_relay`, przejść przez system do odbiorcy podanego w polu klucza. Gdy używane jest słowo kluczowe `HATER`, `check_mail` i `check_relay` są stosowane tylko względem odbiorców, dla których to słowo figuruje we wpisie do bazy danych. `FRIEND` i `HATER` nie mogą jednocześnie pojawić się w tej samej bazie danych *access*, ponieważ funkcja *delay_checks* musi zostać skonfigurowana do przyjmowania albo słowa kluczowego `FRIEND`, albo `HATER` — nie może akceptować jednocześnie obu. Receptura 6.13 przedstawia przykład użycia tych słów kluczowych.

Większość akcji pokazanych w tabeli 6.1 została opisana jako wpływające na wiadomości „do lub spod" adresu. Jest to prawdą tylko wtedy, gdy używane są znaczniki lub funkcja *blacklist_recipients*. Gdy funkcja ta nie jest używana, opisane akcje wpływają jedynie na wiadomości pochodzące z adresu źródłowego, chyba że pole adresu jest zmodyfikowane przez opcjonalny znacznik `To:`, `From:` lub `Connect:`. Znaczniki te ograniczają test adresu odpowiednio do odbiorcy koperty, do nadawcy koperty i do adresu połączenia. Na przykład wpis w bazie danych *access* odrzucający połączenia z 10.0.187.215 może zawierać:

```
Connect:10.0.187.215      ERROR:5.7.1:550 Nie przyjmujemy spamu
```

Znacznik `Connect:` ogranicza dopasowanie do adresu zdalnego systemu, który połączył się z serwerem w celu doręczenia poczty. Jeśli adresem tym będzie 10.0.187.215, wiadomość zostanie odrzucona i do nadawcy zostanie zwrócony komunikat „Nie przyjmujemy UCE".

Adres w polu klucza wpisu w bazie danych *access* może definiować użytkownika, pojedynczy adres e-mail, źródłowy adres IP, adres sieciowy lub nazwę domeny:

- Osoba jest definiowana z użyciem albo pełnego adresu e-mail w postaci *użytkownik@host.domena*, albo nazwy użytkownika w postaci *nazwaużytkownika@*.
- Host jest identyfikowany przez nazwę lub adres IP.
- Domena jest identyfikowana przez nazwę domeny.
- Sieć jest identyfikowana przez składnik sieci w adresie IP.

[2] Rozdział 5. zawiera dodatkowe informacje o kodach DSN i słowach kluczowych kodów SMTP. Patrz też RFC 2821.

Oprócz powyższego przykładu `Connect:` w bazie danych *access* do identyfikacji poczty z 10.0.187.215 mogą zostać użyte inne formaty. Oto kilka z nich:

```
10.0.187                REJECT
[10.0.187.215]          DISCARD
example.com             ERROR:5.7.1:550 Wiadomość odrzucona.
```

Dwa pierwsze wiersze w tej bazie danych *access* dopasowują adresy IP. Pierwszy wpis odrzuca wiadomości z każdego komputera, którego adres IP zaczyna się od numeru sieci 10.0.187, z której pochodziły niepożądane reklamy. Drugi wiersz definiuje konkretny komputer o adresie 10.0.187.215. Nawiasy prostokątne obejmujące pojedynczy adres oznaczają, że ten adres IP nie rozwiązuje się na nazwę hosta.

Ostatni wpis definiuje całą domenę, powodując odrzucenie poczty z każdego hosta w domenie *example.com*. Oczywiście nie użylibyśmy tej metody, gdybyśmy otrzymali tylko jedną wiadomość z reklamą z tej domeny, lecz gdyby nasz system regularnie odbierał z niej spam, moglibyśmy zablokować wszelką pocztę pochodzącą z tej domeny, dopóki jej administratorzy nie poprawiliby bezpieczeństwa. Funkcja *dnsbl* udostępnia kolejną metodę blokowania poczty z określonych hostów i domen.

Czarne listy dla dnsbl i enhdnsbl

Dodanie funkcji *dnsbl* do konfiguracji sendmaila pozwala użyć list źródeł spamu, tzw. *czarnych list* do blokowania spamu. Są to bazy danych DNS identyfikujące źródła przyczyniające się do rozsyłania spamu. Źródłem takim może być zarówno oryginalny nadawca spamu, jak i otwarty przekaźnik poczty pozwalający spamerom rozsyłać wiadomości. Usługi czarnych list są implementowane poprzez DNS. Każdy system uniksowy może wysyłać zapytania DNS, więc jest to bardzo skuteczny sposób dystrybucji informacji. Oczywiście program może wykorzystać informacje tylko wtedy, gdy je rozumie tak jak sendmail.

Funkcja *dnsbl* przyjmuje dwa opcjonalne argumenty. Pierwszym jest nazwa domeny zawierającej listę. Domyślnym ustawieniem jest Realtime Blackhole List (RBL) utrzymywana przez Mail Abuse Prevention System (MAPS). Kilka innych grup również utrzymuje i udostępnia publicznie takie listy. Tabela 6.2 zawiera kilka z nich.

Tabela 6.2. Usługi czarnych list

Nazwa usługi	**Strona WWW**	**Nazwa domeny**
Spamhaus Block List	*www.spamhaus.org*	*sbl.spamhaus.org*
Relay Stop List	*relays.visi.com*	*relays.visi.com*
Distributed Server Boycott List	*dsbl.org*	*list.dsbl.org*
MAPS RBL	*mail-abuse.org*	Niepotrzebna. To jest domyślne ustawienie *dnsbl*.

Aby użyć czarnej listy z tabeli 6.2, należy w pierwszym argumencie *dnsbl* wskazać domenę podaną w tabeli. Na przykład następujące polecenie skonfiguruje użycie przez sendmail Spamhaus Block List:

```
FEATURE(`dnsbl', `sbl.spamhaus.org')
```

Zasady wymuszane przez różne czarne listy mogą być odmienne. Większość z tych usług koncentruje się na blokowaniu otwartych przekaźników, a nie na źródłach spamu. Powodem tego jest fakt, że źródła spamu nieustannie się zmieniają i ukrywają swoje prawdziwe tożsamości. Pomagają im w tym, chociaż niechcący, otwarte przekaźniki. Otwarty przekaźnik poczty zwykle nie chce pomagać spamerom. Zablokowanie poczty z otwartego przekaźnika szybko przyciąga uwagę jego administratora, który poprawia ustawienia systemu i przez to blokuje spamerowi dostęp do zasobów. Taka pośrednia metoda obrony przed spamem sprawia problemy wielu niewinnym, chociaż naiwnym, użytkownikom sieci. Z tego powodu czarne listy są często uważane za lekarstwo równie szkodliwe jak sama choroba, i wiele osób odradza korzystanie z takich list.

Na publicznych czarnych listach znajduje się wiele systemów. Każdy ośrodek, który przekazuje spam — może być nim nawet nasz system, jeśli nie skonfigurujemy poprawnie sendmaila — ma dużą szansę znaleźć się na czarnej liście. Z tego powodu ważna jest poprawna konfiguracja przekazywania. Pomyłka w konfiguracji przekaźnika może spowodować, że nasz serwer wyląduje na czarnej liście. Gdy ośrodek kończy z przekazywaniem spamu, powinien zostać usunięty z listy po okresie około miesiąca. Oczywiście te zasady są różne dla różnych list, podobnie jak skuteczność, z jaką adresy są dodawane i usuwane z listy. Jeśli nasz adres zostanie dodany na czarną listę, powinniśmy rozwiązać problem i zgłosić adres do usunięcia z listy, postępując zgodnie z instrukcjami dostępnymi w serwisie WWW listy. Zanim zaczniemy korzystać z usług czarnych list, powinniśmy odwiedzić serwisy WWW każdej z nich i dowiedzieć się czegoś więcej o liście.

Najprostszym sposobem blokowania spamu jest pozostawienie tego komuś innemu. Z drugiej strony, chociaż korzystanie z publicznych czarnych list jest proste, to rozwiązanie nie jest doskonałe. Nie możemy wybierać adresów dodawanych do listy, co oznacza, że lista może blokować pocztę z „uczciwego" serwera tylko dlatego, że jego administrator zapomniał wyłączyć przekazywanie. Możemy jednak ignorować poszczególne wpisy z czarnej listy za pomocą bazy danych *access* (patrz receptura 6.4). Aby osiągnąć jeszcze dokładniejszą kontrolę, niektóre organizacje tworzą własne czarne listy oparte na usłudze DNS. Receptura 6.5 pokazuje, jak możemy zbudować i wykorzystać własną czarną listę.

Drugim argumentem dostępnym w funkcji *dnsbl* jest komunikat o błędzie wyświetlany, gdy wiadomość zostaje odrzucona na podstawie informacji z serwera czarnej listy. Format domyślnego komunikatu wygląda następująco:

```
550 Rejected $&{client_addr} listed at dnsbl-domain
```

gdzie `%&{client_addr}` jest adresem IP, który został odrzucony, a *`domena-dnsbl`* oznacza czarną listę DNS, która odrzuciła adres (tzn. *`domena-dnsbl`* jest wartością z pierwszego argumentu podanego dla funkcji *dnsbl*). Drugiego argumentu *dnsbl* możemy użyć tylko wtedy, gdy chcemy zmienić standardowy komunikat błędu. Większość administratorów pozostaje przy pierwszym komunikacie.

Trzeci argument dostępny dla funkcji *dnsbl* pozwala określić, jak będą traktowane przejściowe błędy w wyszukiwaniu DNS. Domyślnie sendmail nie wstrzymuje wiadomości tylko dlatego, że usługa czarnej listy nie potrafi odpowiedzieć na wyszukiwanie przez

DNS. Umieszczenie `t` w polu trzeciego argumentu spowoduje, że sendmail będzie zwracał komunikat o przejściowym problemie i wstrzyma wiadomości. Oto przykład:

```
FEATURE(`dnsbl', `sbl.spamhaus.org', ,`t')
```

Alternatywą dla funkcji *dnsbl* jest funkcja *enhdnsbl*. Składnia funkcji *enhdnsbl* ma trzy pierwsze argumenty takie same jak *dnsbl*, lecz dodaje czwarty argument, którym jest wartość zwracana, jakiej sendmail spodziewa się z wyszukiwania przez DNS. Domyślnie dowolna wartość zwracana przez wyszukiwanie przez DNS w usłudze czarnej listy wskazuje, że sprawdzany adres znajduje się na liście usługi i powinien zostać odrzucony. Czwarty argument pozwala zmienić to zachowanie tak, że tylko wartość pasująca do niego będzie powodowała odrzucenie wiadomości. Argumentem tym nie musi być pojedyncza wartość: może to być lista wartości lub takie same operatory jak po lewej stronie reguły przekształceń, pasujące do wielu wartości. Oto przykład funkcji *enhdnsbl*:

```
FEATURE(`enhdnsbl', `sbl.spamhaus.org', , ,`127.0.0.2', `127.0.0.3.')
```

Ten wpis makra włącza funkcję *enhdnsbl* i używa usługi czarnej listy *sbl.spamhaus.org*. Zgodnie z nim sendmail będzie odrzucał przychodzące wiadomości, gdy usługa czarnej listy w odpowiedzi na wyszukiwanie adresu połączenia zwróci wartość 127.0.0.2 lub 127.0.0.3.

Baza danych *access* i czarne listy blokują pocztę ze znanych źródeł spamu i z otwartych przekaźników. Lecz nie wszystkie niechciane reklamy pochodzą ze znanych źródeł spamu. Czasem dopiero po przeczytaniu zorientujemy się, że mamy do czynienia z taką pocztą. Narzędzia filtrujące pocztę mogą sprawdzać zawartość wiadomości i na jej podstawie decydować, jak wiadomość ma zostać potraktowana.

MILTER

sendmail udostępnia poprzez interfejs gniazd bezpośredni dostęp do zewnętrznych programów filtrujących pocztę, zwanych MILTER i napisanych zgodnie z *Sendmail Mail Filter API*. Zewnętrzne filtry poczty są definiowane w konfiguracji sendmaila za pomocą makr `INPUT_MAIL_FILTER` lub `MAIL_FILTER`. Poza nazwą składnia obu makr jest identyczna. Na przykład składnia makra `INPUT_MAIL_FILTER` wygląda następująco:

```
INPUT_MAIL_FILTER(nazwa, przyrównania)
```

`nazwa` jest arbitralną nazwą używaną przez sendmail, podobnie jak wewnętrzne nazwy dostarczycieli poczty i baz danych. W składni znajdują się maksymalnie trzy przyrównania zapisane w postaci *`litera=wartość`*, gdzie literą może być:

`S`

Przyrównanie `S` jest wymagane, ponieważ definiuje gniazdo służące do komunikacji z zewnętrznym filtrem. Definicja gniazda jest zapisana w formie `S=`*`typ`*`:`*`specyfikacja`*, gdzie typ oznacza typ gniazda, a specyfikacja definiuje gniazdo w sposób wymagany dla danego typu gniazda. Obsługiwane są trzy typy gniazd:

`S=unix:`*`ścieżka`*

Gniazda uniksowe są obsługiwane; *`ścieżka`* oznacza pełną ścieżkę do takiego gniazda. Słowo kluczowe `unix` może być zastąpione synonimem `local` (np. `S=local:/var/run/filter1.sock`).

`S=inet:port@host`

Słowo kluczowe `inet` żąda gniazda IP. *`port`* oznacza numer portu sieciowego używanego przez filtr. *`host`* jest nazwą hosta lub adresem IP systemu, w którym filtr jest uruchomiony.

`S=inet6:port@host`

Gniazda IPv6 również są obsługiwane. Słowo kluczowe `inet6` żąda gniazda IPv6, *`port`* identyfikuje numer portu sieciowego używanego przez filtr, a *`host`* system, w którym filtr jest uruchomiony. Jako wartość host może zostać zapisany adres IPv6 lub nazwa hosta odwzorowująca się na adres IPv6.

`F=litera`

To opcjonalne przyrównanie pozwala zdefiniować sposób reakcji na błąd gniazda. Domyślnie w razie awarii gniazda lub nieoczekiwanej odpowiedzi filtru sendmail kontynuuje przetwarzanie wiadomości. Przyrównanie `F=R` pozwala w przypadku problemów z gniazdem lub filtrem odrzucić połączenie z trwałym błędem, a `F=T` z błędem tymczasowym.

`T=litera:wartość;litera:wartość;...`

Użycie tego opcjonalnego przyrównania pozwala zmienić domyślne czasy oczekiwania. Dostępne są cztery wartości literowe:

`C`

Definiuje czas oczekiwania na połączenie. Domyślnie czas oczekiwania upływa, gdy nie można nawiązać połączenia przez 5 minut (`5m`).

`E`

Definiuje ogólny czas oczekiwania, domyślnie 5 minut (`5m`).

`R`

Definiuje czas oczekiwania na odczyt odpowiedzi z filtru, domyślnie 10 sekund (`10s`).

`S`

Definiuje czas oczekiwania przy wysyłaniu danych do filtru, domyślnie 10 sekund (`10s`).

Przy takiej składni makro `INPUT_MAIL_FILTER` dodające obsługę zewnętrznego programu *MIMEDefang* może wyglądać następująco:

```
INPUT_MAIL_FILTER(`mimedefang',`S=unix:/var/run/mimedefang.sock, T=S:5m;R:5m')
```

Powyższe makro definiuje dla tego filtru wewnętrzną nazwę `mimedefang`. sendmail utworzy gniazdo */var/run/mimedefang.sock* i będzie komunikować się z filtrem przez to gniazdo uniksowe. Ponieważ sendmail tworzy gniazdo, nie powinno ono uprzednio istnieć. W powyższym przykładzie nie zostało użyte przyrównanie `F`, więc sendmail będzie standardowo przetwarzać wiadomości nawet w przypadku błędu gniazda lub nieprawidłowej odpowiedzi filtru. Przyrównanie `T` zwiększa liczniki czasu dla wysyłania i odbioru do pięciu minut.

Makro `INPUT_MAIL_FILTER` definiuje tylko jeden filtr. Aby użyć większej liczby filtrów, możemy dodać do konfiguracji sendmaila większą liczbę makr `INPUT_MAIL_FILTER` lub `MAIL_FILTER`. Gdy używamy wielu filtrów, różnica między `INPUT_MAIL_FILTER` i `MAIL_FILTER` staje się widoczna. Załóżmy na przykład, że konfiguracja sendmaila zawiera następujące makra:

```
INPUT_MAIL_FILTER(`filtr1', `S=unix:/var/run/filtr1.soc')
INPUT_MAIL_FILTER(`filtr2', `S=unix:/var/run/filtr2.soc')
INPUT_MAIL_FILTER(`filtr3', `S=unix:/var/run/filtr3.soc')
```

Makro `INPUT_MAIL_FILTER` określa kolejność, w której filtry będą używane. Biorąc pod uwagę trzy powyższe makra i kolejność ich zapisania, sendmail będzie przesyłać dane przez `filtr1`, `filtr2` i `filtr3` w tej właśnie kolejności. Aby utworzyć równoważną konfigurację za pomocą makr `MAIL_FILTER`, potrzebne będą cztery wiersze w konfiguracji sendmaila:

```
MAIL_FILTER(`filtr1', `S=unix:/var/run/filtr1.soc')
MAIL_FILTER(`filtr2', `S=unix:/var/run/filtr2.soc')
MAIL_FILTER(`filtr3', `S=unix:/var/run/filtr3.soc')
define(`confINPUT_MAIL_FILTERS', `filtr1, filtr2, filtr3')
```

Makra `MAIL_FILTER` nie ustalają kolejności użycia filtrów, więc musi im towarzyszyć definicja `confINPUT_MAIL_FILTERS` określająca kolejność wykonania. Gdyby definicja `conf➥INPUT_MAIL_FILTERS` nie została użyta razem z makrami `MAIL_FILTER`, filtry zdefiniowane przez te makra byłyby ignorowane. Oczywiście przy użyciu makr `MAIL_FILTER` i definicji `confINPUT_MAIL_FILTERS` filtry pocztowe nie muszą być stosowane w kolejności ich deklarowania. Na przykład, zmieniając definicję `confINPUT_MAIL_FILTERS` na poniższą, uruchomimy filtry w odwrotnej kolejności:

```
define(`confINPUT_MAIL_FILTERS', `filtr3, filtr2, filtr1')
```

Filtry używane przez sendmail są programami zewnętrznymi. Każdy w miarę doświadczony programista może napisać prosty program filtrujący pocztę, lecz stworzenie filtru, który naprawdę skutecznie będzie zwalczał nadużycia poczty, stanowi poważne wyzwanie. Na szczęście wielu dobrych programistów napisało już przydatne filtry. Aby nie wyważać otwartych drzwi, najlepiej będzie, jeśli najpierw poszukamy w internecie filtrów, które mogą rozwiązać nasze problemy z pocztą. Oto kilka miejsc, od których możemy zacząć poszukiwania:

http://www.milter.org/

Ogólne źródło informacji o programach MILTER.

http://www.mimedefang.org/

Potężny i rozszerzalny MILTER obsługujący skanowanie w poszukiwaniu wirusów, usuwanie załączników na podstawie nazwy i zawartości oraz przetwarzanie Spam ➥Assassin[3].

[3] Informacje o programie SpamAssassin znajdziemy pod adresem *http://spamassassin.org/*.

http://www.snert.com/Software/milter-sender/

MILTER próbujący weryfikować, czy adres nadawcy jest prawdziwy. Wielu spamerów nie używa prawdziwych adresów nadawcy.

http://www.amavis.org/

MILTER skanujący w poszukiwaniu wirusów.

http://sendmail.com/

Sendmail, Inc. udostępnia szereg komercyjnych programów MILTER.

Istnieje wiele innych witryn WWW związanych z programami MILTER i wiele innych takich programów. Jednakże nie są one jedynymi narzędziami dostępnymi do filtrowania poczty — powszechnie stosowany jest również *procmail*.

Filtrowanie za pomocą programu procmail

Większość tekstów dotyczących sendmaila (a niniejszy nie jest wyjątkiem) w filtrowaniu poczty koncentruje się na programie procmail. Istnieje ku temu kilka powodów:

- *procmail* jest ściśle zintegrowany z sendmailem.
- *procmail* jest potężnym narzędziem, które może posłużyć do wielu innych zastosowań poza filtrowaniem spamu.
- *procmail* jest domyślnym programem doręczającym lokalne wiadomości w systemie Linux i może być używany jako dostarczyciel poczty `local` w każdym systemie uniksowym, jeśli dodamy do konfiguracji sendmaila funkcję *local_procmail*.
- Polecenie `MAILER(procmail)` dodaje *procmail* do listy dostarczycieli poczty sendmaila.

Różnorodność sposobów wywoływania programu *procmail* zwiększa elastyczność tego narzędzia. Jak wspomniano powyżej, możemy skonfigurować sendmail do korzystania z procmaila w roli dostarczyciela poczty `local`. procmail może być też uruchomiony z poziomu wiersza poleceń powłoki, z *mailertable*, jak w recepturze 6.8, oraz z pliku *.forward* użytkownika, jak poniżej:

```
$ cat > .forward
"|/usr/bin/procmail"
Ctrl-D
```

Powszechnie spotykanym błędem jest myślenie, że jeśli filtry poczty o zasięgu całego systemu wpływają na dużą liczbę użytkowników, to najważniejsze filtrowanie wiadomości odbywa się na poziomie systemu. Filtrowanie poczty na poziomie użytkownika jest równie ważne. Filtry użytkownika:

- Stanowią ostatnią linię obrony przed spamem, który przedostaje się przez filtry systemu.
- Pozwalają użytkownikom egzekwować własne zasady poczty elektronicznej.

- Pozwalają na filtrowanie na podstawie zawartości bez obawy o naruszenie prywatności.
- Mają do przetworzenia znacznie mniejsze objętości poczty.

Z tych i innych powodów warto zachęcać użytkowników do poznania i korzystania z dostępnych dla nich filtrów poczty. Wiele narzędzi pocztowych przeznaczonych dla użytkowników końcowych zawiera funkcje filtrowania poczty. Nie są one zintegrowane z sendmailem, więc nie będą tu omawiane.

procmail jest potężnym, aczkolwiek złożonym systemem filtrowania poczty. Osobiste filtry procmaila są definiowane przez użytkownika w jego katalogu macierzystym w pliku o nazwie *.procmailrc*. Administrator systemu definiuje filtry poczty obowiązujące dla całego systemu w pliku */etc/procmailrc* i wykorzystuje ten plik do ogólnego filtrowania spamu. Użytkownik końcowy za pomocą pliku *.procmailrc* dodaje własne filtrowanie zgodnie ze swoimi preferencjami. Format obu plików jest taki sam.

Plik *.procmailrc* zawiera dwa typy wpisów — przypisania zmiennych środowiskowych i reguły filtrowania poczty, które w żargonie procmaila noszą nazwę *regułek* (dosłownie *receptur — recipe*). Przypisania zmiennych środowiskowych są oczywiste i wyglądają tak samo jak w skryptach powłoki. Na przykład `HOME=/home/craig` jest poprawnym przypisaniem zmiennej środowiskowej. Dokumentacja man pliku *.procmailrc* wymienia ponad 30 zmiennych środowiskowych.

Główną treścią pliku *.procmailrc* są regułki. Składnia każdej regułki wygląda następująco:

```
:0 [znaczniki] [:[lockfile]]
[* warunek]
akcja
```

Każda regułka zaczyna się od `:0`, co odróżnia ją od instrukcji przypisania. Po `:0` opcjonalnie następują znaczniki zmieniające przetwarzanie przez filtr. Tabela 6.3 wymienia znaczniki i ich zastosowania.

Opcjonalna zmienna *`lockfile`* pozwala podać nazwę lokalnego pliku blokady, który będzie użyty dla tej regułki. Plik blokady zapobiega jednoczesnemu zapisywaniu do tej samej skrzynki pocztowej przez większą liczbę kopii programu procmail, co może zdarzyć się w intensywnie używanym systemie. Nazwę pliku blokady poprzedza dwukropek. Jeśli użyjemy dwukropka bez podania nazwy pliku, zostanie użyta domyślna nazwa *`lockfile`* utworzona z nazwy skrzynki pocztowej z rozszerzeniem *.lock*. Jeśli nie podamy żadnego lokalnego pliku blokady, zostanie użyty plik domyślny, jednakże dokumentacja *procmaila* zaleca korzystanie z lokalnych plików blokady.

Test warunku jest opcjonalny. Jeśli *`warunek`* nie zostanie podany, regułka zadziała tak, jakby warunek był prawdziwy, co oznacza podjęcie działania. Jeśli podajemy warunek, musimy zacząć go od znaku gwiazdki (`*`). warunek jest zapisywany w postaci wyrażenia regularnego. Jeśli wartość zdefiniowana przez wyrażenie regularne zostanie znaleziona w wiadomości, warunek zostaje rozwiązany jako prawda i akcja zostaje podjęta. Aby podjąć akcję, gdy wiadomość *nie* zawiera podanej wartości, wyrażenie regularne musi zaczynać się od wykrzyknika. Oto dwa przykłady poprawnych testów warunku:

Tabela 6.3. Znaczniki regułek procmaila

Znacznik	Znaczenie
A	Wykonaj tę regułkę, jeśli poprzednia zwróciła wartość prawda.
a	To samo znaczenie co znacznik A, lecz dodatkowo poprzedzająca regułka musi zostać z powodzeniem wykonana.
b	Przekaż treść wiadomości do miejsca przeznaczenia. Opcja domyślna.
B	Filtruj treść wiadomości.
c	Utwórz kopię DW tej wiadomości.
D	Testy rozróżniają wielkość liter. Domyślnie wielkość liter jest ignorowana.
e	Wykonaj tę regułkę, jeśli wykonanie poprzedniej receptury zwróciło błąd.
E	Wykonaj tę regułkę, jeśli poprzednia nie była wykonana.
f	Prześlij dane przez zewnętrzny program filtrujący.
H	Filtruj nagłówki wiadomości. Opcja domyślna.
h	Przekaż nagłówek wiadomości do miejsca przeznaczenia. Opcja domyślna.
I	Ignoruj błędy zapisu w tej regułce.
r	Zapisz wiadomość bez dodatkowego sprawdzania formatu.
w	Sprawdź kod zakończenia zewnętrznego programu filtrującego.
W	To samo co znacznik w, lecz komunikat o błędzie nie jest wysyłany na wyjście.

```
* ^From.*simon@oreilly\.com
* !^Subject: Chapter
```

Pierwszy warunek sprawdza, czy wiadomość zawiera wiersz zaczynający się (^) od ciągu znaków From, po którym następuje dowolna liczba znaków (.*) i ciąg simon@oreilly.com. Drugi warunek pasuje do wszystkich wiadomości niezawierających (!) wiersza zaczynającego się od łańcucha Subject: Chapter. Jeśli w jednej regułce zdefiniowana jest większa liczba warunków, każdy warunek powinien znaleźć się w osobnym wierszu.

Wprawdzie regułka *procmaila* może zawierać wiele warunków, lecz tylko jedną akcję. Akcja może przekierować wiadomość do pliku, przekazać dalej pod inny adres e-mail, wysłać do programu lub zdefiniować dodatkowe regułki, które będą przetwarzać wiadomość. Jeśli akcja jest regułką dodatkową, to zaczyna się od :0. Jeśli kieruje pod adres e-mail, powinna zacząć się od wykrzyknika (!), a jeśli kieruje do programu, zaczyna się od symbolu pionowej poprzeczki (|). Jeśli akcja wysyła wiadomość do pliku, podawana jest tylko nazwa tego pliku. Poniższy przykład ilustruje, jak poczta jest przekazywana do przetworzenia przez zewnętrzny program:

```
:0 B
* .*pheromones
| awk -f spamscript > spam-suspects
```

Znacznik B stosuje test warunku do zawartości listu. Wszystkie wiadomości zawierające słowo „pheromones" w dowolnym miejscu tekstu są przekazywane w celu przetworzenia do *awk*. W tym przykładzie *awk* uruchamia plik programu o nazwie *spamscript*, który

wydobywa informacje z wiadomości i zapisuje w pliku o nazwie *spam-suspects*. Możemy przypuszczać, że administrator tego systemu napisał *spamscript*, aby wydobywać adresy e-mail z przypuszczalnego spamu.

Powyższy przykład przedstawia filtrowanie przez *procmail* całej treści wiadomości. Domyślnie *procmail* sprawdza tylko nagłówki wiadomości. Nagłówkom możemy poświęcić szczególną uwagę w konfiguracji sendmaila, używając niestandardowych zestawów reguł.

Niestandardowe zestawy reguł

sendmail pozwala definiować niestandardowe przetwarzanie adresów i nagłówków przychodzących wiadomości i udostępnia do tego celu kilka punktów zaczepienia. Do niestandardowego przetwarzania adresów służą:

`Local_check_relay`
: To jest punkt zaczepienia do zestawu reguł `check_relay`. Do zestawu reguł `Local_check_relay` jest przekazywana nazwa hosta i adres IP hosta, który zainicjował połączenie e-mail.

`Local_check_mail`
: To jest punkt zaczepienia do zestawu reguł `check_mail`, który przetwarza adres nadawcy koperty z polecenia SMTP `MAIL From:`. Receptura 6.10 zawiera przykładowy zestaw reguł `Local_check_mail`.

`Local_check_rcpt`
: To jest punkt zaczepienia do zestawu reguł `check_rcpt`, który sprawdza adres odbiorcy koperty zdefiniowany poleceniem SMTP `RCPT To:`.

Te zestawy reguł nie zostały zaprojektowane tylko do wykrywania i usuwania spamu; mają szersze zastosowanie. Jednakże te punkty zaczepień zestawów reguł są przydatne do zwalczania spamu.

Oprócz tych punktów zaczepienia, wywoływanych ze standardowych zestawów reguł, możemy wywołać zestaw reguł z definicji nagłówka, aby dokonać własnego przetwarzania nagłówka. Podstawowa składnia polecenia `H` pliku *sendmail.cf* definiuje format nagłówków poczty. W podstawowej składni definicja nagłówka zaczyna się od polecenia `H`, po którym następuje nazwa nagłówka i jego format. Składnia wywołania zestawu reguł z polecenia `H` wygląda następująco:

```
Hnazwa: $>zestawreguł
```

gdzie *`nazwa`* oznacza nazwę nagłówka, a *`zestawreguł`* jest zestawem reguł wywoływanym do przetworzenia przychodzących nagłówków o tej nazwie.

Ta funkcjonalność może posłużyć do sprawdzania przychodzących nagłówków w celu wykrywania spamu na podstawie informacji w nagłówku. Receptura 6.9 zawiera przykład wykorzystania tej metody.

6.1. Blokowanie spamu za pomocą bazy danych access

Problem

Ponieważ większość wiadomości przychodzących z konkretnych lokalizacji to spam, należy skonfigurować sendmail tak, by blokował całość poczty z tych miejsc.

Rozwiązanie

Dodaj adresy, które chcesz zablokować, do pliku tekstowego */etc/mail/access*. Kluczem w każdym wpisie jest adres spamera, a wartością zwracaną `DISCARD` — aby odrzucić wiadomość „po cichu", `REJECT` — aby odrzucić wiadomość ze standardowym błędem, lub `ERROR` — aby odrzucić wiadomość z nietypowym komunikatem o błędzie. Za pomocą *makemap* zbuduj z pliku tekstowego bazę danych typu hash.

Następnie utwórz konfigurację sendmaila zawierającą funkcję *access_db*. Oto wymagane makro `FEATURE`:

```
dnl Użyj bazy danych access
FEATURE(`access_db')
```

Podobnie jak w przykładzie z receptury 1.8 zrekompiluj plik *sendmail.cf*, skopiuj nowy plik *sendmail.cf* do */etc/mail* i uruchom ponownie sendmail.

Analiza

W bazie danych *access* `REJECT`, `ERROR` i `DISCARD` mogą posłużyć do blokowania niechcianej poczty. Poniższy przykład wpisów w bazie danych *access* blokuje wiadomości z trzech miejsc:

```
example.com          REJECT
wrotethebook.net     ERROR:5.7.1:550 Niepoprawne źródło wiadomości
fake.ora.com         DISCARD
```

Test *telnet* pokaże, co te zdalne ośrodki zobaczą w zależności od akcji zdefiniowanych w bazie danych:

```
# telnet localhost smtp
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 chef.wrotethebook.com ESMTP Sendmail 8.12.9/8.12.9;
Fri, 22 Aug 2003 12:01:37 -0400
HELO localhost
250 chef.wrotethebook.com Hello
IDENT:UWSRv+Jij66J8vALUBVBECbGPVoU8OQe@localhost [127.0.0.1],
pleased to meet you
MAIL From:<crooks@example.com>
550 5.7.1 <crooks@example.com>... Access denied
```

```
MAIL From:<thieves@wrotethebook.net>
550 5.7.1 <thieves@wrotethebook.net>... Niepoprawne źródło wiadomości
MAIL From:<junk@fake.ora.com>
250 2.1.0 <junk@fake.ora.com>... Sender ok
QUIT
221 2.0.0 chef.wrotethebook.com closing connection
Connection closed by foreign host.
```

Wiadomość z *example.com* została odrzucona z komunikatem o błędzie „Access denied", ponieważ wpis dla *example.com* w przykładowej bazie danych *access* definiuje dla wiadomości z tej domeny akcję `REJECT`. Wiadomość z *wrotethebook.net* została odrzucona z komunikatem o błędzie „Niewłaściwe źródło wiadomości", który został zdefiniowany w bazie danych *access* za pomocą polecenia `ERROR`. Dla odmiany z punktu widzenia zdalnego systemu wiadomość z *fake.ora.com* wydaje się zostać przyjęta przez serwer. Aby zobaczyć działanie akcji `DISCARD` zdefiniowanej w bazie danych *access* dla *fake.ora.com*, potrzebny jest test `sendmail -bt`:

```
# sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> check_mail junk@fake.ora.com
check_mail          input: junk @ fake . ora . com
Basic_check_mail    input: junk @ fake . ora . com
tls_client          input: $| MAIL
D                   input: < > < ? > < ! "TLS_Clt" > < >
D                 returns: < ? > < > < ? > < ! "TLS_Clt" > < >
A                   input: < > < ? > < ! "TLS_Clt" > < >
A                 returns: < > < ? > < ! "TLS_Clt" > < >
TLS_connection      input: $| < > < ? > < ! "TLS_Clt" > < >
TLS_connection    returns: OK
tls_client        returns: OK
CanonAddr           input: < junk @ fake . ora . com >
canonify            input: < junk @ fake . ora . com >
Canonify2           input: junk < @ fake . ora . com >
Canonify2         returns: junk < @ fake . ora . com >
canonify          returns: junk < @ fake . ora . com >
Parse0              input: junk < @ fake . ora . com >
Parse0            returns: junk < @ fake . ora . com >
CanonAddr         returns: junk < @ fake . ora . com >
SearchList          input: < + From > $| < F : junk @ fake . ora . com >
< U : junk @ > < D : fake . ora . com > < >
F                   input: < junk @ fake . ora . com > < ? > < + From > < >
F                 returns: < ? > < >
SearchList          input: < + From > $| < U : junk @ >
< D : fake . ora . com > < >
U                   input: < junk @ > < ? > < + From > < >
U                 returns: < ? > < >
SearchList          input: < + From > $| < D : fake . ora . com > < >
D                   input: < fake . ora . com > < ? > < + From > < >
D                 returns: < DISCARD > < >
SearchList        returns: < DISCARD >
SearchList        returns: < DISCARD >
SearchList        returns: < DISCARD >
Basic_check_mail returns: $# discard $: discard
check_mail        returns: $# discard $: discard
> /quit
```

W tym teście adres *junk@fake.ora.com* jest przetwarzany przez zestaw reguł `check_mail`, który sprawdza adres `MAIL From:`. Ten zestaw reguł przetwarza adres i zwraca „discard", co oznacza, że poczta z *fake.ora.com* będzie odrzucona bez informowania o tym nadawcy.

Zobacz również

Plik *cf/README*, rozdział 3. niniejszej książki i wstęp do niniejszego rozdziału dostarczają dodatkowych informacji o bazie danych *access*. Książka *sendmail* omawia bazę danych *access* w podrozdziale 7.5.

6.2. Uniemożliwienie lokalnym użytkownikom odpowiedzi na spam

Problem

Niektórzy lokalni użytkownicy odpowiadają na listy ze spamem, zachęcając przez to spamerów do dalszego działania. Chcemy skonfigurować sendmail tak, aby powstrzymywał spamerów i osoby „wspierające" spam.

Rozwiązanie

Przed założeniem jakiegokolwiek konta użytkownika zdefiniuj zasady dopuszczalnego użytku które, między innymi, dadzą prawo blokowania wiadomości związanych ze spamem — zarówno przychodzących, jak i wychodzących. Upewnij się, że wszyscy użytkownicy zgodzą się na te zasady, zanim przyznasz komukolwiek konto użytkownika.

Dodaj adresy źródeł spamu, które chcesz zablokować, do pliku tekstowego */etc/mail/access*. Użyj znaczników `To:` i `From:`, aby uniemożliwić wysyłanie poczty do spamerów i akceptowanie wiadomości od spamerów. Uruchom *makemap*, aby zbudować bazę danych typu hash z pliku tekstowego.

Utwórz konfigurację sendmaila, która za pomocą funkcji *access_db* włącza korzystanie z bazy danych *access*. Wymagane polecenie `FEATURE` wygląda następująco:

```
dnl Użyj bazy danych access
FEATURE(`access_db')
```

Zrekompiluj plik *sendmail.cf*, skopiuj nowy plik *sendmail.cf* do */etc/mail* i ponownie uruchom sendmail.

Analiza

Domyślnie baza danych *access* stosuje się do adresów źródłowych. Akcje zdefiniowane we wpisach w bazie danych są podejmowane na podstawie źródła wiadomości. W przypadku bazy danych *access* z receptury 6.1 odrzucane są wiadomości z *example.com*, *wrotethebook.net* i *fake.ora.com*, co pokazują testy w tej recepturze. Na przykład wiadomości od wszelkich

użytkowników z *example.com* są odrzucane z błędem „Access denied". Jednakże baza danych *accesss* z receptury 6.1 nie zapobiega wysyłaniu poczty z lokalnego hosta do domeny *example.com*.

Dodanie znacznika `To:` do wpisu w bazie danych *access* powoduje zastosowanie akcji zdefiniowanej we wpisie do adresu odbiorcy pasującego do klucza, natomiast znacznik `From:` żąda zastosowania akcji do pasującego adresu źródłowego. Baza danych *access* z receptury 6.1 zmodyfikowana z użyciem znaczników `To:` i `From:` wygląda tak:

```
From:example.com          REJECT
To:example.com            ERROR:5.7.1:550 Wiadomości pod ten adres
nie są dozwolone
From:wrotethebook.net     ERROR:5.7.1:550 Niewłaściwe źródło wiadomości
To:wrotethebook.net       ERROR:5.7.1:550 Wiadomości pod ten adres
nie są dozwolone
From:fake.ora.com         DISCARD
To:fake.ora.com           ERROR:5.7.1:550 Wiadomości pod ten adres
nie są dozwolone
```

Ponieważ dla wpisu `From:` *example.com* akcją jest *REJECT*, wiadomości z tej domeny są odrzucane, jak widać w recepturze 6.1. Po dodaniu wpisu `To:` wiadomości zaadresowane do *example.com* również są odrzucane:

```
# telnet localhost smtp
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 chef.wrotethebook.com ESMTP Sendmail 8.12.9/8.12.9;
Fri, 22 Aug 2003 12:01:37 -0400
HELO localhost
250 chef.wrotethebook.com Hello
IDENT:UWSRv+Jij66J8vALUBVBECbGPVoU8OQe@localhost [127.0.0.1],
pleased to meet you
MAIL From:<craig@chef.wrotethebook.com>
250 2.1.0 <craig@chef.wrotethebook.com>... Sender ok
RCPT To:<crook@example.com>
550 5.7.1 <crook@example.com>... Wiadomości pod ten adres nie są dozwolone
QUIT
221 2.0.0 chef.wrotethebook.com closing connection
Connection closed by foreign host.
```

Przy blokowaniu wychodzącej poczty należy szczególnie uważać. Lokalni użytkownicy oczekują, że będą mogli komunikować się z kimkolwiek i nie chcą, abyśmy za nich decydowali, z kim mogą rozmawiać, a z kim nie. Zasady dopuszczalnego użytkowania, które dają nam takie uprawnienia, muszą powstać, zanim podejmiemy jakiekolwiek działania. I niezależnie od treści tego dokumentu powinniśmy przygotować się na skargi.

Rozwiązania alternatywne

Funkcja *blacklist_recipients* jest alternatywną metodą blokowania wiadomości wysyłanych do znanych spamerów. Funkcja *blacklist_recipients* stosuje każdy wpis bez znacznika w bazie danych *access* do adresów odbiorców. Poniższe wpisy dodane do konfiguracji sendmaila włączają obsługę bazy danych *access* i stosują ją do adresów odbiorców:

```
dnl Użyj bazy danych access
FEATURE(`access_db')
dnl Zastosuj bazę access również do adresów odbiorców
FEATURE(`blacklist_recipients')
```

Funkcja *blacklist_recipients* jest skuteczna i bardzo łatwa w użyciu. Ponieważ jednak stosuje się do wszystkich wpisów w bazie danych *access* niezawierających znaczników, to nie jest dostępna w niej kontrola nad konfiguracją umożliwiana przez znacznik `To:`. Oprócz tego znaczniki stanowią dla samych siebie dokumentację. Każdy, kto spojrzy na przykładową bazę danych *access* przedstawioną powyżej, widząc znacznik `To:` i tekst błędu w polu akcji, zrozumie, że wiadomości wysyłane do *example.com* nie są akceptowane.

Zobacz również

Rozdział 3. i wstęp do niniejszego rozdziału zawierają dodatkowe informacje o bazie danych *access*. Książka *sendmail* omawia bazę danych *access* w podrozdziale 7.5 i funkcję *blacklist_recipients* w punkcie 7.5.5. Sekcja *Anti-Spam Configuration Control* pliku *cf/README* również omawia ten temat.

6.3. Odczytywanie bazy danych access z LDAP

Problem

Należy tak skonfigurować sendmail, by odczytywał bazę danych *access* z serwera LDAP.

Rozwiązanie

W razie potrzeby zrekompiluj i zainstaluj ponownie sendmail, aby dodać obsługę LDAP do sendmaila, oraz dodaj schemat sendmaila do konfiguracji LDAP w serwerze LDAP. Oba kroki zostały opisane w recepturze 1.3.

W serwerze LDAP wprowadź rekordy bazy danych *access*, używając formatu klasy obiektu `sendmailMTAMap` zdefiniowanej w schemacie sendmaila. Za pomocą skryptu *ldapadd* zapisz rekordy *access* w bazie danych LDAP.

W hoście sendmaila dodaj do konfiguracji sendmaila definicję `confLDAP_CLUSTER` i funkcję *access_db*. Ustaw w `confLDAP_CLUSTER` tę samą wartość co w atrybucie `sendmailMTAClu➥ster` rekordów *access* bazy danych LDAP. Dodaj łańcuch `LDAP` do polecenia `FEATURE` *access_db*, aby sendmail czytał dane *access* z LDAP. Oto przykładowe wpisy, które można dodać do konfiguracji sendmaila:

```
dnl Definicja nazwy klastra LDAP
define(`confLDAP_CLUSTER', `wrotethebook.com')
dnl Czytaj bazę danych access z LDAP
FEATURE(`access_db', `LDAP')
```

Zrekompiluj i zainstaluj ponownie plik *sendmail.cf*, a następnie uruchom ponownie sendmail. Przykład znajduje się w recepturze 1.8.

Analiza

Dystrybucja sendmaila zawiera plik schematu LDAP definiujący podstawowe atrybuty niezbędne dla baz danych i klas sendmaila. Możemy też oczywiście zdefiniować własny schemat, lecz użycie domyślnego upraszcza konfigurację zarówno usługi LDAP, jak i sendmaila. Przy użyciu schematu sendmaila do definiowania wpisów *access* w bazie danych LDAP poniższy przykład przekształca wpisy *access*, użyte w recepturze 6.1, na rekordy LDAP:

```
# cat > ldap-access
dn: sendmailMTAMapName=access, dc=wrotethebook, dc=com
objectClass: sendmailMTA
objectClass: sendmailMTAMap
sendmailMTACluster: wrotethebook.com
sendmailMTAMapName: access

dn: sendmailMTAKey=example.com, sendmailMTAMapName=access, dc=wrotethebook,
dc=com
objectClass: sendmailMTA
objectClass: sendmailMTAMap
objectClass: sendmailMTAMapObject
sendmailMTAMapName: access
sendmailMTACluster: wrotethebook.com
sendmailMTAKey: example.com
sendmailMTAMapValue: REJECT

dn: sendmailMTAKey=wrotethebook.net, sendmailMTAMapName=access, dc=wrotethebook,
dc=com
objectClass: sendmailMTA
objectClass: sendmailMTAMap
objectClass: sendmailMTAMapObject
sendmailMTAMapName: access
sendmailMTACluster: wrotethebook.com
sendmailMTAKey: wrotethebook.net
sendmailMTAMapValue: ERROR:5.7.1:550 Niepoprawne źródło wiadomości

dn: sendmailMTAKey=fake.ora.com, sendmailMTAMapName=access, dc=wrotethebook,
dc=com
objectClass: sendmailMTA
objectClass: sendmailMTAMap
objectClass: sendmailMTAMapObject
sendmailMTAMapName: access
sendmailMTACluster: wrotethebook.com
sendmailMTAKey: fake.ora.com
sendmailMTAMapValue: DISCARD
Ctrl-D
# ldapadd -x -D "cn=Manager,dc=wrotethebook,dc=com" \
> -W -f ldap-access
Enter LDAP Password: tajnehasloLDAP
adding new entry "sendmailMTAMapName=access, dc=wrotethebook, dc=com"

adding new entry "sendmailMTAKey=example.com, sendmailMTAMapName=access,
dc=wrotethebook, dc=com"
```

```
adding new entry "sendmailMTAKey=wrotethebook.net, sendmailMTAMapName=access,
dc=wrotethebook, dc=com"

adding new entry "sendmailMTAKey=fake.ora.com, sendmailMTAMapName=access,
dc=wrotethebook, dc=com"
```

Aby dodać trzy wpisy *access* z receptury 6.1, potrzebne były cztery rekordy LDAP. Pierwszy z nich nadaje w LDAP nazwę mapy bazy danych *access*. Kolejne rekordy LDAP używają tej nazwy przy dodawaniu rekordów *access* do bazy danych LDAP.

Następne trzy rekordy definiują trzy wpisy *access* opisane w recepturze 6.1. Proszę zwrócić uwagę, że atrybuty `sendmailMTAKey` i `sendmailMTAMapValue` każdego rekordu odpowiadają parom klucz-wartość z oryginalnych wpisów w *access*. Zmieniając wartości zapisane w atrybutach `sendmailMTAMapName`, `sendmailMTAKey` i `sendmailMTAMapValue`, możemy użyć podstawowego formatu rekordu LDAP stosowanego do bazy danych *access* do każdej innej bazy danych sendmaila.

Po skonwertowaniu tych rekordów z pliku LDIF i dodaniu ich do bazy danych LDAP możemy je sprawdzić poleceniem *ldapsearch*:

```
# ldapsearch -LLL -x '(sendmailMTAMapName=access)' sendmailMTAMapValue
dn: sendmailMTAMapName=access, dc=wrotethebook, dc=com

dn: sendmailMTAKey=example.com, sendmailMTAMapName=access, dc=wrotethebook,
dc=com
sendmailMTAMapValue: REJECT

dn: sendmailMTAKey=wrotethebook.net, sendmailMTAMapName=access, dc=wrotethebook,
dc=com
sendmailMTAMapValue: ERROR:5.7.1:550 Niepoprawne źródło wiadomości

dn: sendmailMTAKey=fake.ora.com, sendmailMTAMapName=access, dc=wrotethebook,
dc=com
sendmailMTAMapValue: DISCARD
```

Ten test pokazuje, że rekordy bazy danych *access* są dostępne w serwerze LDAP. Jeśli nasz system sendmail wymaga wartości `-h` i `-b` w teście *ldapsearch*, te same wartości będą wymagane w konfiguracji sendmaila. Parametry `-h` i `-b` ustawiamy za pomocą definicji `confLDAP_DEFAULT_SPEC`, jak w recepturze 5.9.

Teraz musimy skonfigurować sendmail tak, by korzystał z serwera LDAP. Najpierw dodaliśmy polecenie `confLDAP_CLUSTER` do konfiguracji sendmaila, aby podać dla sendmaila nazwę klastra LDAP. Schemat sendmaila dopuszcza rekordy stosujące się do pojedynczych hostów lub do grup hostów zwanych *klastrami* (*cluster*). Rekordy LDAP stosujące się do pojedynczych hostów używają atrybutu `sendmailMTAHost`. sendmail pobiera tylko rekordy używające atrybutu `sendmailMTAHost`, jeśli wartość przypisana do tego atrybutu jest pełną złożoną nazwą hosta sendmaila. Rekordy stosujące się do grup hostów używają atrybutu `sendmailMTACluster`. Aby pobierać rekordy używające tego atrybutu, w sendmailu musi zostać skonfigurowana nazwa klastra, dokładnie jak w niniejszej recepturze. W tej recepturze definiujemy rekordy *access* w usłudze LDAP za pomocą atrybutu `sendmailMTACluster` i informujemy sendmail o nazwie klastra przez definicję `confLDAP_CLUSTER`.

Dodanie argumentu `LDAP` do polecenia `FEATURE` *access_db* powoduje, że sendmail będzie czytał bazę danych *access* z serwera LDAP, używając standardowego schematu sendmaila. Jeśli zdefiniujemy własny schemat, będziemy musieli poinformować sendmail, jak powinien go używać do pobierania rekordów *access*. Na przykład:

```
FEATURE(`access_db', `ldap: -1 -k (&(objectClass=OurAccessDB)
(OurAccesDBKey=%0)) -v OurAccessDBValue')
```

Przykładowe nazwy atrybutów powinny zostać zignorowane, jednakże format powyższego polecenia `FEATURE` jest podobny do tego, które będziemy musieli zdefiniować, aby pobierać dane *access* za pomocą własnego schematu LDAP. Opcja `-k` definiuje kryteria wyszukiwania LDAP używane jako klucz bazy danych. Atrybuty użyte w tych kryteriach wyszukiwania powinny być zgodne z atrybutami zdefiniowanymi w naszym schemacie. Opcja `-v` określa atrybut LDAP zawierający wartość zwracaną. Ponownie musi on zgadzać się z atrybutem naszego niestandardowego schematu. Użycie domyślnego schematu sendmaila upraszcza konfigurację programu sendmail. Wystarczy użyć łańcucha `LDAP` w poleceniu `FEATURE` *access_db*, jak w rozwiązaniu.

Kilka testów przeprowadzonych po zainstalowaniu rozwiązania z niniejszej receptury pokazuje, że sendmail czyta dane LDAP. Najpierw wykonamy test `sendmail -bt` i użyjemy polecenia `/map` do pobrania rekordu *access* z rekordu LDAP:

```
# sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> /map access fake.ora.com
map_lookup: access (fake.ora.com) returns DISCARD (0)
> /quit
```

Ten test pokazuje, że baza danych *access* funkcjonuje tak samo niezależnie od tego, czy jest czytana z lokalnej bazy danych czy z serwera LDAP. Ponowne wykonanie testu z receptury 6.1 pokazuje, że sendmail blokuje wiadomości z użyciem LDAP dokładnie tak samo jak z użyciem lokalnej bazy danych *access*:

```
# sendmail -bs
220 rodent.wrotethebook.com ESMTP Sendmail 8.12.9/8.12.9;
Thu, 27 Mar 2003 12:42:41 -0500
MAIL From:<crooks@example.com>
550 5.7.1 <crooks@example.com>... Access denied
MAIL From:<thieves@wrotethebook.net>
550 5.7.1 <thieves@wrotethebook.net>... Niepoprawne źródło wiadomości
QUIT
221 2.0.0 rodent.wrotethebook.com closing connection
```

LDAP nie zmienia sposobu działania sendmaila. Decyzja o użyciu LDAP nie bierze się z wymogów sendmaila, lecz jest motywowana usługą LDAP. Jeśli stosujemy już LDAP do centralizacji zarządzania informacjami, możemy też dodać dane konfiguracyjne do naszego serwera LDAP.

Zobacz również

Receptury 6.1 i 6.2 opisują sposoby użycia bazy danych *access* do kontroli spamu; receptura 6.1 opisuje konkretne wpisy w bazie danych *access* użyte w niniejszej recepturze. Plik *cf/README* omawia ten temat w sekcji *Using LDAP for Aliases, Maps and Classes*. Książka *sendmail* omawia funkcję *access_db* w podrozdziale 7.5 i definicję `confLDAP_CLUSTER` w podpunkcie 21.9.82.

6.4. Korzystanie z usługi czarnej listy DNS

Problem

Należy skonfigurować sendmail tak, by korzystał z usługi czarnej listy w celu blokady dużych ilości spamu z wielu źródeł przy minimalnych nakładach pracy.

Rozwiązanie

Dodaj funkcję *dnsbl* do konfiguracji sendmaila. Zidentyfikuj konkretną usługę czarnej listy, której chcesz użyć w wierszu polecenia *dnsbl*. Oto przykład:

```
dnl Użyj usługi czarnej listy DSBL
FEATURE(`dnsbl', `list.dsbl.org')
```

Posługując się recepturą 1.8, zrekompiluj plik *sendmail.cf*, skopiuj nowy plik *sendmail.cf* do */etc/mail* i uruchom ponownie sendmail.

Analiza

Funkcja *dnsbl* dodaje kod sendmail.cf niezbędny, aby włączyć usługę czarnej listy (ang. *blacklist*) DNS. Funkcja *dnsbl* używa polecenia K do zdefiniowania bazy danych *dnsbl* jako bazy danych typu `host`, co oznacza, że wyszukiwanie w *dnsbl* w rzeczywistości jest przekazywane w celu rozwiązania do usługi DNS[4]. Funkcja *dnsbl* dodaje też kilka reguł do zestawu reguł `Basic_check_relay` wywoływanego z zestawu reguł `check_relay`. Dodane reguły wyszukują adres połączenia w bazie danych *dnsbl*. Jeśli adres ten zostanie znaleziony w bazie danych, wiadomość z niego pochodząca jest odrzucana z komunikatem o błędzie. Jeśli adres połączenia nie zostanie znaleziony w bazie *dnsbl*, wiadomość jest przekazywana do dalszego przetwarzania. Test `sendmail -bt` pokazuje działanie dodanych reguł przekształceń:

```
# sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
```

[4] Zakładamy, że plik zamiany usług mapuje wyszukiwanie hosta na DNS. Dodatkowe informacje o pliku zamiany usług zawiera rozdział 5.

```
> .D{client_addr}192.168.111.68
> Basic_check_relay <>
Basic_check_rela   input: < >
Basic_check_rela returns: $# error $@ 5 . 7 . 1 $:
"550 Rejected: " 192 . 168 . 111 . 68 " listed at list.dsbl.org"
> /quit
```

Ponieważ nie istnieje aktywne połączenie — to tylko test — pierwszym krokiem jest statyczne zdefiniowanie adresu połączenia dla tego testu. Następnie wywoływany jest zestaw reguł `Basic_check_relay` i zostaje przekazany do pustego obszaru roboczego. Obszar roboczy przekazany do zestawu reguł w tym teście jest nieistotny, ponieważ pierwsza reguła dodana do zestawu reguł przez funkcję *dnsbl* bezwarunkowo zastępuje obszar roboczy wartością wziętą z `${client_addr}`. Wobec tego wartością wyszukiwaną w bazie danych *dnsbl* jest adres połączenia zapisany w makrze `${client_addr}`. W tym teście adres 192.168.111.68 został znaleziony na czarnej liście utrzymywanej w *list.dsbl.org*, więc wiadomość z tego adresu została odrzucona z komunikatem o błędzie:

```
550 Rejected: 192.168.111.68 listed at list.dsbl.org
```

Komunikat o błędzie podaje odrzucony adres i usługę, która zaleciła odrzucenie. Ta informacja jest ważna. Administratorzy spod adresu `192.168.111.68` mogą chcieć skontaktować się z tą usługą, aby dowiedzieć się, czemu ich system znalazł się na czarnej liście oraz co mogą zrobić, aby został z niej usunięty. Często system trafia na czarną listę z powodu błędu w konfiguracji, który tworzy otwarty przekaźnik. Natychmiast po naprawieniu problemu administrator chciałby, aby jego system został usunięty z czarnej listy. Wiedza, które usługi wciągnęły system na czarną listę, pozwala administratorowi skontaktować się z tymi usługami, aby jego usługi pocztowe zostały w pełni przywrócone.

W tej konfiguracji został użyty serwer czarnej listy pod adresem *list.dsbl.org*, ponieważ tę właśnie usługę podaliśmy w poleceniu `FEATURE` funkcji *dnsbl* w niniejszej recepturze. Jest to tylko przykład, a nie zalecenie korzystania z usługi *list.dsbl.org*. Dostępnych jest wiele usług czarnych list, z których część została wymieniona w tabeli 6.2. Radzimy odwiedzić witryny WWW wszystkich tych usług i zapoznać się z ich zasadami wpisywania hostów do bazy danych, a następnie wybrać tę usługę, której zasady najlepiej pasują do zasad, jakie chcemy egzekwować w swoim serwerze.

Gdy w wierszu polecenia *dnsbl* nie jest podana żadna usługa, sendmail domyślnie używa *blackholes.mail-abuse.org* — tej samej usługi, która była używana przez wycofaną funkcję sendmaila, *rbl*.

W niniejszej recepturze mogliśmy też użyć funkcji *enhdnsbl*, lecz w tym konkretnym przypadku *enhdnsbl* nie przynosi większych korzyści.

Zobacz również

Przed skorzystaniem z dowolnej usługi czarnej listy warto odwiedzić jej witrynę WWW. Użycie takiej usługi powoduje, że zewnętrzna organizacja decyduje o tym, które wiadomości nasz system będzie odbierać. Zasady i misja usługi czarnej listy powinny zostać

ocenione pod kątem zgodności z naszymi potrzebami. Receptury 6.5 i 6.6 zawierają dodatkowe informacje o usługach czarnych list, które warto przejrzeć przed zaimplementowaniem niniejszej receptury. Książka *sendmail* omawia funkcję *dnsbl* w punkcie 7.2.1 i *enhdnsbl* w punkcie 7.2.2.

6.5. Tworzenie własnej czarnej listy DNS

Problem

Chcemy utworzyć własną czarną listę DNS, ponieważ żadna zewnętrzna usługa tego typu nie udostępnia dokładnie takich zasad wpisywania na listę i zawartości listy, jaka jest nam potrzebna.

Rozwiązanie

Administrator domeny musi utworzyć plik strefy DNS w poprawnym formacie, zawierający wszystkie adresy połączeń, które mają być blokowane. Specjalne rekordy adresów DNS w pliku strefy są tworzone przez odwrócenie adresu IP systemu wpisanego na czarną listę, aby utworzyć pole nazwy DNS w rekordzie, oraz przez użycie adresu typu `127.0.0.2` w polu danych rekordu adresu. Taki format oznacza, że hosty są wpisywane na czarną listę według adresów IP, a nie nazw, co jest rozsądnym rozwiązaniem, ponieważ wyszukiwanie *dnsbl* odbywa się na podstawie adresu IP połączenia. Serwer DNS musi być autorytatywny dla domeny, w której czarna lista ma się mieścić. Zwykle do tego celu tworzy się specjalną poddomenę dla czarnej listy w obrębie upoważnień serwera DNS.

W systemie sendmaila utwórz konfigurację zawierającą funkcję *dnsbl*[5]. Zidentyfikuj lokalną czarną listę w wierszu polecenia *dnsbl*, na przykład:

```
dnl Skieruj dnsbl do naszej lokalnej czarnej listy DNS
FEATURE(`dnsbl', `dnsbl.wrotethebook.com')
```

Posługując się recepturą 1.8, zrekompiluj plik *sendmail.cf*, skopiuj nowy plik *sendmail.cf* do */etc/mail* i uruchom ponownie sendmail.

Analiza

Korzystanie z usługi czarnej listy (ang. *blackhole*) jest proste, lecz mało elastyczne, ponieważ nie decydujemy o tym, które adresy znajdą się na liście. Oznacza to, że wiadomości z przyjaznego serwisu mogą być blokowane tylko dlatego, że jego administrator źle skonfigurował przekazywanie. Z tego powodu niektóre organizacje decydują się tworzyć własne czarne listy oparte na DNS-ie. Utworzenie własnego serwera czarnej listy gwarantuje,

[5] Może być też użyta tutaj funkcja *enhdnsbl*.

że łączność z wszystkimi adresami, z którymi chcemy się kontaktować, pozostaje pod naszą bezpośrednią kontrolą, lecz wymaga ekspertyzy zarówno z dziedziny sendmaila, jak i DNS-u.

Administrator DNS używa instrukcji `zone` w pliku *named.conf* serwera DNS do ładowania bazy danych czarnej listy. Zakładając, że czarna lista hostów została zdefiniowana w pliku strefy o nazwie *blacklisted.hosts*, który zawiera dane dla domeny o nazwie *dnsbl.wrotethebook.com*, użyta będzie następująca instrukcja:

```
zone "dnsbl.wrotethebook.com" IN {
        type master;
        file "blacklisted.hosts";
        allow-update { none; };
};
```

Wpisy na czarnej liście dla adresów `10.0.187.215` i `192.168.0.3` będą w pliku *blacklisted.hosts* zdefiniowane następująco:

```
215.187.0.10                IN A 127.0.0.2
3.0.168.192                 IN A 127.0.0.2
```

Utworzona właśnie domena DNS jest wskazywana jako źródło danych czarnej listy w wierszu polecenia funkcji *dnsbl* w rozwiązaniu. Wiadomości z każdej domeny wymienionej w domenie *dnsbl.wrotethebook.com* są odrzucane, jak widać z tej próby wysłania wiadomości z `192.168.0.3`:

```
# telnet chef smtp
Trying 192.168.0.8...
Connected to 192.168.0.8.
Escape character is '^]'.
220 chef.wrotethebook.com ESMTP Sendmail 8.12.9/8.12.9;
Fri, 22 Aug 2003 12:01:37 -0400
helo rodent.wrotethebook.com
250 chef.wrotethebook.com Hello rodent.wrotethebook.com [192.168.0.3],
pleased to meet you
MAIL From:<craig@rodent.wrotethebook.com>
550 5.7.1 Rejected: 192.168.0.3 listed at dnsbl.wrotethebook.com
QUIT
221 2.0.0 chef.wrotethebook.com closing connection
Connection closed by foreign host.
```

Adres połączenia `192.168 .0.3` został znaleziony w domenie *dnsbl.wrotethebook.com*, więc nasz serwer odrzucił wiadomość i zwrócił komunikat o błędzie „550 5.7.1 Rejected: 192.168.0.3 listed at dnsbl.wrotethebook.com.”. Ten domyślny komunikat o błędzie możemy zmienić za pomocą dodatkowego argumentu wiersza polecenia w funkcji *dnsbl*. Na przykład, polecenie:

```
FEATURE(`dnsbl', `dnsbl.wrotethebook.com', `" Wiadomość odrzucona.
"$&{client_addr}" jest podejrzany o przekazywanie spamu."')
```

zmienia komunikat na „Wiadomość odrzucona. `192.168.0.3` jest podejrzany o przekazywanie spamu.” Jednakże standardowy komunikat jest wystarczająco dobry i daje zdalnemu serwerowi więcej informacji.

Niewielka liczba systemów umieszczonych na czarnej liście w niniejszej recepturze mogłaby być znacznie łatwiej obsłużona przez bazę danych *access*. W większości przypadków użycie bazy danych *access* do blokowania niechcianych połączeń pocztowych jest o wiele łatwiejsze niż tworzenie własnej czarnej listy. Tworzenie i utrzymywanie własnej czarnej listy wymaga sporych nakładów pracy. Systemy, które trzeba dodawać i usuwać z listy, nieustannie się zmieniają. Oprócz tego do wstępnego zbudowania listy potrzebna jest duża objętość informacji. Możemy użyć narzędzia filtrującego pocztę, takiego jak *procmail*, do automatycznego gromadzenia podejrzanych adresów prosto z wiadomości. Jednakże trudno jest stworzyć system, który gromadzi właściwe informacje i utrzymuje je aktualne. Większość administratorów woli tworzyć własne czarne listy w oparciu o czarne listy udostępniane przez odpowiednie usługi. Serwisy te dysponują już dużymi listami, które nieustannie utrzymują. Większość usług czarnych list udostępnia jakieś metody pobrania całej listy. Na przykład listę DSBL możemy pobrać za pomocą *rsync*:

```
# rsync rsync.dsbl.org::dsbl/bind-list.dsbl.org .
```

Okresowe pobieranie listy i dostosowywanie jej do własnych potrzeb jest jednym ze sposobów na utworzenie własnej czarnej listy. Lecz zanim zaczniemy od gotowej listy, musimy pamiętać, że tworzenie własnej czarnej listy nie jest zadaniem, które można podjąć niefrasobliwie. Jest to jedna z najtrudniejszych technik kontroli nad niechcianym spamem.

Jak zwykle wybór między użyciem usługi czarnej listy i stworzeniem własnej jest wyborem pomiędzy prostotą a elastycznością. Większość ośrodków wybiera prostotę. Jeśli nie dysponujemy wystarczającymi siłami, aby zbudować i utrzymywać własną czarną listę, najlepiej będzie, gdy pozostaniemy przy publicznej usłudze czarnej listy, jak w recepturze 6.4, i, korzystając z receptury 6.6, dostosujemy ją do swoich potrzeb.

Zobacz również

Receptura 6.4 opisuje prostą metodę korzystania z czarnej listy. Receptura 6.6 opisuje, jak można uporać się z głównymi niedostatkami usług czarnej listy, używając bazy danych *access*. Obie receptury warto przejrzeć przed zaimplementowaniem niniejszej. Książka *sendmail* omawia funkcję *dnsbl* w punkcie 7.2.1 i *enhdnsbl* w punkcie 7.2.2. Informacje o konfiguracji DNS dostępne są w książce *DNS and BIND* Paula Albitza i Cricketa Liu (O'Reilly) oraz *Linux DNS Server Administration* Craiga Hunta (Sybex).

6.6. Usuwanie adresów z czarnej listy

Problem

Używamy usługi czarnej listy zawierającej kilka adresów, z którymi musimy się komunikować. Należy tak skonfigurować sendmail, by dla konkretnych adresów ignorował czarną listę.

Rozwiązanie

Chcąc zignorować usługę czarnej listy dla konkretnego adresu, dodaj ten adres do pliku tekstowego */etc/mail/access* i przydziel jako wartość zwracaną dla tego adresu słowo kluczowe `OK`. Za pomocą *makemap* utwórz z pliku tekstowego bazę danych typu hash.

Utwórz konfigurację sendmaila, która używa funkcji *dnsbl* lub *enhdnsbl* do wybrania usługi czarnej listy i *access_db* do ignorowania czarnej listy w przypadku wybranych adresów. Oto przykładowe wpisy, które można dodać do konfiguracji sendmaila, aby włączyć te funkcje:

```
dnl Użyj dnsbl i wybierz usługę czarnej listy
FEATURE(`dnsbl', `list.dsbl.org')
dnl Używaj bazy danych access
FEATURE(`access_db')
```

Zrekompiluj plik *sendmail.cf* i skopiuj do */etc/mail*, a następnie uruchom ponownie sendmail (patrz receptura 1.8).

Analiza

Funkcja *dnsbl* dodaje do konfiguracji *sendmail.cf* obsługę czarnej listy DNS i wskazuje usługę, która będzie używana. Tabela 6.2 zawiera listę kilku z dostępnych serwisów. Jeśli nie zostanie wybrana konkretna usługa, domyślnie zostanie użyta MAPS RBL. Usługę powinniśmy wybrać z rozwagą.

Przykładowa czarna lista blokuje wiadomości z 192.168.0.3, co pokazuje poniższy test wykonany z 192.168.0.3:

```
# telnet chef smtp
Trying 192.168.0.8...
Connected to 192.168.0.8.
Escape character is '^]'.
220 chef.wrotethebook.com ESMTP Sendmail 8.12.9/8.12.9;
Fri, 22 Aug 2003 12:01:37 -0400
helo rodent.wrotethebook.com
250 chef.wrotethebook.com Hello rodent.wrotethebook.com [192.168.0.3],
pleased to meet you
MAIL From:<craig@rodent.wrotethebook.com>
550 5.7.1 Rejected: 192.168.0.3 listed at list.dsbl.org
QUIT
221 2.0.0 chef.wrotethebook.com closing connection
Connection closed by foreign host.
```

Baza danych *access* pozwala zignorować wybrane wpisy w bazie danych czarnej listy. W tym przykładzie zignorujemy usługę czarnej listy dla następujących adresów:

```
# cd /etc/mail
# cat > access
192.168.0.3        OK
24.199.249.90      OK
Ctrl-D
# makemap hash access < access
```

Po skompilowaniu bazy danych *access* ponowne uruchomienie testu *telnet* z 192.168.0.3 da następujący wynik:

```
# telnet chef smtp
Trying 192.168.0.8...
Connected to 192.168.0.8.
Escape character is '^]'.
220 chef.wrotethebook.com ESMTP Sendmail 8.12.9/8.12.9;
Fri, 22 Aug 2003 12:01:37 -0400
helo rodent.wrotethebook.com
250 chef.wrotethebook.com Hello rodent.wrotethebook.com [192.168.0.3],
pleased to meet you
MAIL From:<craig@rodent.wrotethebook.com>
250 2.1.0 <craig@rodent.wrotethebook.com>... Sender ok
QUIT
221 2.0.0 chef.wrotethebook.com closing connection
Connection closed by foreign host.
```

Teraz poczta z 192.168.0.3 jest przyjmowana, mimo że adres ten nadal znajduje się na czarnej liście, ponieważ w bazie danych *access* akcją zdefiniowaną dla 192.168.0.3 jest `OK`.

Dzięki wykorzystaniu bazy danych *access* do ignorowania konkretnych wpisów z usługi czarnej listy tworzenie własnej czarnej listy dla większości organizacji staje się zbędne. Ogólnie mówiąc, organizacje cofają się przed korzystaniem z usług czarnych list, ponieważ te mogą blokować pocztę z przyjaznych adresów. Połączenie bazy danych *access* z usługą czarnej listy daje nam prostotę usługi czarnej listy i elastyczność bezpośredniej kontroli nad tym, z którymi adresami będziemy się komunikować. Oprócz tego, jeśli usługa czarnej listy nie zawiera adresu, który naszym zdaniem powinien być blokowany, możemy zablokować taki adres, używając bazy danych *access* jak w recepturze 6.1.

Zobacz również

Receptury 6.4 i 6.5 zawierają inne przykłady korzystania z czarnej listy. Receptury 6.1 i 6.2 dostarczają dodatkowych informacji o korzystaniu z bazy danych *access* do kontroli nad spamem. Więcej ogólnych informacji o bazie danych *access* zawiera rozdział 3. Książka *sendmail* omawia funkcję *dnsbl* w punkcie 7.2.1, *enhdnsbl* w punkcie 7.2.2 i funkcję *access_db* w podrozdziale 7.5. Sekcja *Anti-Spam Configuration Control* pliku *cf/README* również porusza te tematy.

6.7. Filtrowanie lokalnej poczty za pomocą procmaila

Problem

Wiadomości należy przefiltrować za pomocą programu procmail przed doręczeniem do lokalnych użytkowników.

Rozwiązanie

Dodaj funkcję *local_procmail* do konfiguracji sendmaila, umieszczając makro `FEATURE` po makrze `OSTYPE` i przed wierszem `MAILER(`local')` w głównym pliku konfiguracyjnym. Wartości dla dostarczyciela poczty `local` są zwykle ustawiane w pliku `OSTYPE`. Dokładnie przejrzyj plik dla swojego systemu operacyjnego. Dodaj funkcję *local_procmail* do głównego pliku konfiguracyjnego tylko wtedy, gdy funkcja ta nie jest zawarta w pliku `OSTYPE`.

Utwórz plik */etc/procmailrc*, zawierający filtry, które chcesz zastosować do poczty lokalnej.

Skompiluj i zainstaluj nową konfigurację, jak w recepturze 1.8.

Analiza

Plik `OSTYPE` *linux.m4* zawiera funkcję *local_procmail*, ponieważ *procmail* jest dostarczycielem poczty `local` używanym domyślnie w większości systemów Linux. W systemie linuksowym uruchomienie konfiguracji używającej pliku `OSTYPE` *linux.m4* wystarczy dla niniejszej receptury. W innych systemach nie jest to tak proste. Załóżmy na przykład, że mamy system Solaris 8. Plik `OSTYPE` *solaris8.m4* używa funkcji *local_lmtp*, aby wybrać `mail.local` do roli dostarczyciela poczty `local`. Aby zmienić go na procmail, musimy przesłonić funkcję *local_lmtp*, umieszczając *local_procmail* w głównym pliku konfiguracyjnym. Oto przykład oparty na pliku *generic-solaris.mc*:

```
VERSIONID(`Solaris with local_procmail added')
OSTYPE(solaris2)
DOMAIN(generic)
dnl Add the local_procmail feature
FEATURE(`local_procmail')
MAILER(local)
MAILER(smtp)
```

Ponieważ funkcja *local_procmail* występuje po makrze `OSTYPE`, zastępuje funkcję *local_lmtp* zdefiniowaną w pliku `OSTYPE`. Funkcja *local_procmail* stanie się aktywna po skompilowaniu pliku *sendmail.cf*, skopiowaniu do */etc/mail* i ponownym uruchomieniu sendmaila.

Gdy używana jest funkcja *local_procmail*, sendmail przekazuje lokalną pocztę do programu *procmail* w celu doręczenia. *procmail* doręcza wiadomości, najpierw używając poleceń zdefiniowanych w pliku */etc/procmailrc*, a następnie poleceń zdefiniowanych w pliku *.procmailrc* w katalogu macierzystym odbiorcy. Jeśli nie został zdefiniowany żaden plik *rc*, procmail zapisuje niezmienioną wiadomość do skrzynki odbiorczej użytkownika. Proszę zwrócić uwagę, że plik *.procmailrc* użytkownika jest stosowany do poczty doręczanej przez funkcję *local_procmail*. Gdy ta funkcja jest wykorzystywana, użytkownik nie musi wywoływać programu procmail z pliku *.forward*, jak było we wstępie do niniejszego rozdziału. Wystarczy, że utworzy plik *.promailrc*, który zostanie zastosowany do jego poczty. Użycie programu *procmail* w roli lokalnego dostarczyciela poczty pozwala zarówno administratorowi, jak i użytkownikowi za jego pomocą filtrować pocztę przychodzącą.

Gdy *procmail* jest używany jako dostarczyciel poczty `local`, sendmail uruchamia go z trzema argumentami: `-Y`, `-a` i `-d`. Opcja `-Y` powoduje używanie przez *procmail* standardowego formatu skrzynki pocztowej Berkeley Unix. Opcja `-d` służy do podania nazwy użytkownika lokalnego odbiorcy, który ma otrzymać pocztę (pośród trzech parametrów doręczenia poczty jest to wartość użytkownika). Opcja `-a` przekazuje do programu *procmail* opcjonalną wartość, która w regułkach *procmaila* jest dostępna jako zmienna `$1`; pośród trzech parametrów doręczenia poczty jest to wartość hosta. sendmail przekazuje wartość przez `-a` tylko wtedy, gdy używana jest składnia `+detal` lub gdy poczta jest kierowana do dostarczyciela poczty `local` przez *mailertable*. W przypadku składni `+detal` przekazywana jest wartość `detal`. W przypadku *mailtertable* przekazywaną wartością jest adres wejściowy, który był kluczem do wpisu w *mailertable*. We wszystkich pozostałych przypadkach przez argumenty `-a` nie jest przekazywana żadna wartość, a zmienna `$1` pozostaje nieprzydzielona.

Funkcja *local_procmail* ma wpływ na bezpieczeństwo dla *smrsh* i prób ograniczenia przekazywania poczty dalej przez użytkowników. Więcej informacji zawierają receptury 10.6 i 10.8.

Zobacz również

Receptura 6.8 dostarcza dodatkowych informacji o programie procmail. Książka *sendmail* omawia *.forward* w rozdziale 13.[6] oraz funkcję *local_procmail* w punkcie 4.8.21[7]. Dodatkowe informacje o filtrowaniu poczty przez *procmail* zawierają strony dokumentacji man dla `procmail`, `procmailrc`, `procmailex` i `procmailsc`.

6.8. Filtrowanie poczty wychodzącej za pomocą procmaila

Problem

Należy tak skonfigurować sendmail, by filtrował wiadomości zaadresowane do konkretnych domen, używając do tego programu *procmail*.

Rozwiązanie

Zbuduj *mailertable* kierującą wiadomości przeznaczone dla konkretnych domen przez dostarczyciela poczty `procmail`.

Utwórz w katalogu */etc/procmailrcs* plik, który definiuje potrzebne opcje filtrowania. Możesz użyć większej liczby filtrów.

[6] Wydanie polskie: *sendmail*, Helion, 2001, podrozdział 25.7.

[7] Wydanie polskie: *sendmail*, Helion, 2001, punkt 19.6.13.

Utwórz konfigurację sendmaila, która włącza funkcję *mailertable* i dodaje *procmail* do listy dostępnych dostarczycieli poczty. Do konfiguracji powinny zostać dodane następujące wpisy:

```
dnl Włącz obsługę mailertable
FEATURE(`mailertable')
dnl Dodaj procmail do listy dostępnych mailerów
MAILER(procmail)
```

Skompiluj plik *sendmail.cf*, skopiuj do */etc/mail/sendmail.cf* i uruchom ponownie sendmail, jak w recepturze 1.8.

Analiza

Makro `MAILER(procmail)` dodaje definicję dostarczyciela poczty `procmail` do pliku *sendmail.cf*. Dostarczyciel poczty `procmail` nie jest związany z funkcją *local_procmail*. System może korzystać z dostarczyciela poczty `procmail` bez używania programu *procmail* jako `local`, a *procmail* może być używany jako `local` bez dodawania do konfiguracji makra `MAILER(procmail)`.

Makro `MAILER(procmail)` nie dodaje do konfiguracji żadnego kodu do używania dostarczyciela poczty `procmail`. Musimy albo dodać własne reguły pliku *sendmail.cf*, aby odwołać się do dostarczyciela poczty, albo kierować pocztę przez dostarczyciela poczty `procmail` za pomocą *mailertable*. Użycie *mailertable* jest najprostszym i zalecanym sposobem dostępu do dostarczyciela poczty. Tutaj dodajemy do *mailertable* wpisy wywołujące *procmail*:

```
# cd /etc/mail
# cat >> mailertable
example.com       procmail:/etc/procmailrcs/spam-filter
wrotethebook.net  procmail:/etc/procmailrcs/spam-filter
fake.ora.com      procmail:/etc/procmailrcs/uce-filter
Ctrl-D
# makemap hash mailertable < mailertable
```

Ten przykład dodaje do *mailertable* trzy wpisy, które kierują pocztę przez dostarczyciela poczty `procmail`. Pierwszym polem wpisu w *mailertable* jest klucz, z którym będzie porównywany adres odbiorcy. Drugie pole to wartość *`mailer`* i wartość *`host`*, których sendmail używa do tworzenia trzech parametrów doręczenia poczty. W tym przykładzie wiadomość z pasującym adresem odbiorcy będzie kierowana przez dostarczyciela poczty `procmail`. Pokaże to kilka testów w systemie z użytą niniejszą recepturą:

```
# sendmail -bv crooks@example.com
crooks@example.com... deliverable: mailer procmail, host /etc/procmailrcs/
spam-filter, user crooks@example.com
# sendmail -bv spammers@wrotethebook.net
spammers@wrotethebook.net... deliverable: mailer procmail,
host /etc/procmailrcs/spam-filter, user spammers@wrotethebook.net
# sendmail -bv thieves@fake.ora.com
thieves@fake.ora.com... deliverable: mailer procmail, host /etc/procmailrcs/
uce-filter, user thieves@fake.ora.com
```

Gdy wiadomość jest kierowana do dostarczyciela poczty procmail, wartość hosta (`$h`) musi zawierać ścieżkę o pliku *rc*, którego procmail powinien użyć do filtrowania poczty. W powyższym przykładzie do programu procmail przekazywane są dwa różne filtry, *spam-filter* i *uce-filter*, zależnie od miejsca przeznaczenia wiadomości. sendmail wywołuje program *procmail* z dostarczyciela poczty `procmail` za pomocą polecenia:

```
procmail -Y -m $h $f $u
```

Znacznik `-Y` mówi, że *procmail* powinien używać formatu skrzynki pocztowej Berkeley Unix. Znacznik `-m` uruchamia procmail jako uniwersalny filtr poczty. Pierwszym argumentem następującym po znaczniku `-m` musi być ścieżka do pliku *rc* zawierającego reguły filtrów *procmaila*. sendmail przypisuje wartość hosta zwróconą przez wyszukiwanie w *mailertable* do makra `$h`, które następnie przekazuje ją do programu *procmail* jako pierwszy argument po znaczniku `-m`. Wobec tego pole *`host`* wpisu w *mailertable* używającego dostarczyciela poczty `procmail` musi zawierać pełną ścieżkę do pliku *rc*.

Następnymi dwoma argumentami przekazanymi do programu *procmail* są adres e-mail nadawcy koperty (`$f`) i adres e-mail odbiorcy koperty (`$u`). Wartości te są dostępne wewnątrz pliku *rc* programu *procmail* odpowiednio jako zmienne `$1` i `$2`.

Filtrowanie poczty wychodzącej za pomocą programu *procmail* umożliwia powstanie pętli kierowania poczty. Regułki usuwające wiadomość, zwracające ją do nadawcy lub przekazujące dalej nie stanowią problemu. Jeśli jednak wiadomość jest sprawdzana, a następnie odsyłana do pierwotnego odbiorcy, wówczas wróci do sendmaila, który skieruje ją do procmaila, a ten wyśle ją do sendmaila, który skieruje… i tak w kółko. Jeśli część wiadomości wychodzących filtrowanych przez procmail będzie odsyłana do pierwotnego odbiorcy, możemy potrzebować dodatkowego kodu w *sendmail.cf*, aby uniknąć pętli.

Często używaną metodą unikania pętli jest dodanie pseudodomeny `.PROCMAIL` do adresu odbiorcy, gdy wiadomość jest z powrotem odsyłana do pierwotnego odbiorcy. Pseudodomena zapewnia, że adres odbiorcy przestaje pasować do wartości w *mailertable*, co przerywa pętlę. Pseudodomenę dodają polecenia *procmaila* w pliku *rc*, jednakże poprawnie skonfigurowany plik *rc* nie stanowi kompletnego rozwiązania. `.PROCMAIL` nie jest prawdziwą domeną, więc do pliku *sendmail.cf* musimy dodać kod zapewniający jej poprawną obsługę. Poniższe makra *m4* i kod *sendmail.cf*, dodane na koniec głównego pliku konfiguracyjnego z niniejszej receptury, obsługują pseudodomenę `.PROCMAIL`, jeśli będzie dodana przez plik *rc*:

```
LOCAL_CONFIG
# Add .PROCMAIL to the pseudo-domain list
CP.PROCMAIL
LOCAL_RULE_0
# Strip .PROCMAIL and send via esmtp
R$+ < @ $+ .PROCMAIL . >        $#esmtp $@ $2 $: $1<@$2>
```

Makro `LOCAL_CONFIG` oznacza początek kodu, który zostanie dodany do sekcji informacji lokalnych pliku *sendmail.cf*. W tym przykładzie dodajemy to sekcji lokalnych informacji komentarz i polecenie `C`. Polecenie to dodaje `.PROCMAIL` do klasy `P`. Klasa `P` zawiera pseu-

dodomeny, których sendmail nie powinien wyszukiwać w DNS-ie. Dodanie `.PROCMAIL` do klasy `P` pozwala uniknąć opóźnień i marnowania zasobów powodowanych przez wyszukiwanie przez sendmail nazwy nieistniejącej domeny.

Makro `LOCAL_RULE_0` oznacza początek kodu *sendmail.cf* dodawanego do zestawu reguł 0, najczęściej nazywanego zestawem reguł `parse`. Dokładniej mówiąc, kod następujący po makrze `LOCAL_RULE_0` jest dodawany do zestawu reguł `ParseLocal`, który jest punktem zaczepienia do zestawu reguł `parse` w miejscu, gdzie dodawane są reguły zdefiniowane lokalnie. Zestaw reguł `parse` przekształca adres doręczenia na trzy parametry doręczenia wiadomości.

Kod następujący po makrze `LOCAL_RULE_0` w tym przykładzie składa się z komentarza i reguły przekształcenia. Polecenie `R` dopasowuje adresy wejściowe w postaci *użytkownik*`@`*domena*`.PROCMAIL` i przekształca na trzy parametry doręczenia poczty, gdzie dostarczycielem poczty jest `esmtp`, wartością hosta *domena*, a wartością użytkownika *użytkownik*`@`*domena*. Po odbudowaniu konfiguracji z użyciem nowego głównego pliku konfiguracyjnego przeprowadzenie testu `sendmail -bv` pokaże działanie tej reguły przekształceń:

```
# sendmail -bv crooks@example.com.PROCMAIL
crooks@example.com.PROCMAIL... deliverable: mailer esmtp, host example.com,
user crooks@example.com
```

Zobacz rónież

Receptura 6.7 zawiera dodatkowe informacje o programie *procmail*. Książka *sendmail* omawia `LOCAL_CONFIG` w podpunkcie 4.3.3.1[8] i `LOCAL_RULE_0` w 4.3.3.2.[9] Dodatkowe informacje o filtrowaniu poczty przez *procmail* zawierają strony dokumentacji man dla `procmail`, `procmailrc`, `procmailex` i `procmailsc`. Receptura 5.1 opisuje *mailertable* i sposób jej użycia do kierowania poczty do dowolnego dostarczyciela poczty o specjalnym przeznaczeniu.

6.9. Wywoływanie specjalnego przetwarzania nagłówków

Problem

Do konfiguracji sendmaila należy dodać niestandardowe kontrole nagłówków.

Rozwiązanie

Dołącz własne przetwarzanie nagłówka na koniec głównego pliku konfiguracyjnego z użyciem makr `LOCAL_CONFIG` i `LOCAL_RULESETS`. Makro `LOCAL_CONFIG` dodaje wiersze do sekcji informacji lokalnych pliku *sendmail.cf*, więc służy do definiowania wszelkich makr,

[8] Wydanie polskie: *sendmail*, Helion, 2001, punkt 19.6.30.

[9] Wydanie polskie: *sendmail*, Helion, 2001, punkt 19.6.32.

klas i baz danych używanych przez nasz niestandardowy proces nagłówka. Za pomocą makra LOCAL_RULESETS dodaj własny zestaw reguł do pliku *sendmail.cf*. Analiza zawiera przykłady użycia obu makr.

Skompiluj plik *sendmail.cf*, skopiuj do */etc/mail/sendmail.cf* i uruchom ponownie sendmail, jak w recepturze 1.8.

Analiza

Plik *knecht.mc* zawarty w dystrybucji sendmaila zawiera różnorodne przykłady ilustrujące różne aspekty konfiguracji programu. Poniższa niestandardowa kontrola nagłówka pochodzi z pliku *knecht.mc*:

```
LOCAL_CONFIG
#
#  Names that won't be allowed in a To: line (local-part and domains)
#
C{RejectToLocalparts}   friend you
C{RejectToDomains}      public.com

LOCAL_RULESETS
HTo: $>CheckTo

SCheckTo
R$={RejectToLocalparts}@$*      $#error $: "553 Header error"
R$*@$={RejectToDomains}         $#error $: "553 Header error"
```

Niestandardowe przetwarzanie nagłówków wymaga, oprócz zwykłej konfiguracji *m4*, dodania kodu do pliku *sendmail.cf*. Makro LOCAL_CONFIG oznacza początek wierszy, które zostaną dodane bezpośrednio do sekcji lokalnych informacji w pliku *sendmail.cf*. W naszym przykładzie dwa wiersze C następujące po LOCAL_CONFIG definiują dwie klasy i ładują do nich pewne wartości.

Makro LOCAL_RULESETS wskazuje, że nastąpi po nim lokalnie zdefiniowany zestaw reguł. Pierwszym wierszem w przykładzie po LOCAL_RULESETS jest polecenie nagłówka, które wywołuje niestandardowy zestaw reguł:

```
HTo: $>CheckTo
```

To polecenie H wywołuje zestaw reguł CheckTo za każdym razem, gdy ze zdalnego systemu w strumieniu danych poczty pojawi się nagłówek To:. Składnia $> jest standardowym sposobem wywoływania zestawów reguł z reguł przekształceń i definicji nagłówków.

Sam zestaw reguł zaczyna się od polecenia S definiującego nazwę zestawu reguł CheckTo. Ten zestaw zawiera dwie reguły przekształceń. Pierwsza reguła dopasowuje każdy nagłówek To: zawierający adres z nazwą użytkownika znajdującą się w klasie $={RejectTo➥LocalParts}. W naszym przykładzie będzie to każda wiadomość zaadresowana do użytkownika o nazwie *friend* lub *you*. Wiadomości zaadresowane do takich użytkowników będą odrzucane z komunikatem błędu „553 Header error”.

Druga reguła przekształcenia dopasowuje każdy nagłówek `To:` adresujący wiadomość do hosta o nazwie obecnej w klasie `$={RejectToDomains}`. W naszym przykładzie klasa ta zawiera tylko nazwę hosta *public.com*. Wiadomości zaadresowane do *public.com* będą odrzucane z komunikatem błędu „553 Header error".

Zestaw reguł `CheckTo` możemy z łatwością przetestować za pomocą `sendmail -bt`:

```
# sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> CheckTo friend@wrotethebook.com
CheckTo            input: friend @ wrotethebook . com
CheckTo          returns: $# error $: "553 Header error"
> CheckTo craig@public.com
CheckTo            input: craig @ public . com
CheckTo          returns: $# error $: "553 Header error"
> CheckTo craig@wrotethebook.com
CheckTo            input: craig @ wrotethebook . com
CheckTo          returns: craig @ wrotethebook . com
> /quit
```

Zestaw reguł jest wywołany i zostaje do niego przekazany przykładowy nagłówek `To:`. Proszę zwrócić uwagę, że do zestawu reguł przekazujemy treść nagłówka bez nazwy nagłówka. Tak będzie wywoływany zestaw reguł podczas faktycznej pracy. We wszystkich trzech testach zestaw reguł `CheckTo` działa zgodnie z oczekiwaniami — nazwa użytkownika *friend* i nazwa hosta *public.com* są odrzucane, lecz adres *craig@wrotethebook.com* przechodzi przez zestaw reguł bez szwanku.

Zestaw reguł `CheckTo` jest tylko uproszczonym przykładem. Zaimplementowanie własnego zestawu reguł przetwarzających nagłówki, który będzie skutecznie zwalczać spam, będzie bardziej złożone. Jednakże każde niestandardowe przetwarzanie nagłówka można zaimplementować z użyciem tej samej składni polecenia `H` wywołującej niestandardowy zestaw reguł nagłówka, tego samego makra `LOCAL_CONFIG` i tego samego makra `LOCAL_RULESETS` co użyte w tym przykładzie. Przed utworzeniem własnego procesu nagłówka w sendmailu, mającego walczyć ze spamem warto zapoznać się z alternatywami, na przykład filtrowaniem poczty za pomocą programów MILTER lub *procmail* i wybrać rozwiązanie najprostsze i najskuteczniejsze w implementacji i utrzymaniu.

Zobacz również

Receptura 6.10 także wykorzystuje przykład z *knecht.mc*. Receptury 6.7 i 6.8 omawiają *procmail*, a receptura 6.12 opisuje MILTER — alternatywy dla przetwarzania nagłówków, które powinniśmy poznać przed rozpoczęciem pisania własnych zestawów reguł *sendmail .cf*. Książka *sendmail* omawia makro `LOCAL_RULESETS` w podpunkcie 4.3.3.5[10] i `LOCAL_CONFIG` w podpunkcie 4.3.3.1[11]. Dodatkowe informacje o poleceniach *sendmail.cf* zawierają książki *TCP/IP Network Administration, Third Edition* Craiga Hunta (O'Reilly) i *Linux Sendmail Administration* Craiga Hunta (Sybex).

[10] Wydanie polskie: *sendmail*, Helion, 2001, punkt 19.6.36.

[11] Wydanie polskie: *sendmail*, Helion, 2001, punkt 19.6.30.

6.10. Używanie w sendmailu wyrażeń regularnych

Problem

sendmail wymaga specjalnej konfiguracji, aby można było w nim wykorzystywać wyrażenia regularne do wyszukiwania wzorców w adresach i nagłówkach.

Rozwiązanie

Uruchom polecenie `sendmail -d01`. Wyjście wiersza „Compiled with:" tego polecenia powinno zawierać `MAP_REGEX`. W przeciwnym razie skompiluj sendmail ponownie, jak w recepturze 1.4.

Dodaj własny kod na koniec głównego pliku konfiguracyjnego. Dodaj polecenie `K`, definiujące wyrażenie regularne, do sekcji informacji lokalnych pliku *sendmail.cf*, używając makra `LOCAL_CONFIG`, oraz za pomocą makra `LOCAL_RULESETS` dodaj niestandardowy zestaw reguł w celu zapewnienia dostępu do wyrażenia regularnego. Analiza zawiera przykłady użycia tych poleceń.

Skompiluj plik *sendmail.cf*, skopiuj go do */etc/mail/sendmail.cf* i uruchom ponownie sendmail, jak w recepturze 1.8.

Analiza

Wyrażenia regularne są definiowane w pliku *sendmail.cf* za pomocą polecenia `K`, które służy również do definiowania baz danych. Następnie dostęp do wyrażenia regularnego w konfiguracji odbywa się tak samo jak dostęp do bazy danych. Poniższy przykład, zaczerpnięty z pliku *knecht.mc* pokazuje sposób definiowania i używania wyrażeń regularnych:

```
LOCAL_CONFIG
#
#  Regular expression to reject:
#    * numeric-only localparts from aol.com and msn.com
#    * localparts starting with a digit from juno.com
#
Kcheckaddress regex -a@MATCH
   ^([0-9]+<@(aol|msn)\.com|[0-9][^<]*<@juno\.com)\.?>

LOCAL_RULESETS
SLocal_check_mail
# check address against various regex checks
R$*                             $: $>Parse0 $>3 $1
R$+                             $: $(checkaddress $1 $)
R@MATCH                         $#error $: "553 Header error"
```

Najpierw do głównego pliku konfiguracyjnego *m4* zostaje dodane makro `LOCAL_CONFIG`. Oznacza ono początek kodu, który ma zostać dodany do sekcji informacji lokalnych pliku *sendmail.cf*. Po tym makrze następuje polecenie `K` definiujące wyrażenie regularne. Składnia polecenia `K` wygląda następująco:

```
Knazwa typ argumenty
```

gdzie `K` jest poleceniem, *`nazwa`* oznacza wewnętrzną nazwę używaną do dostępu do bazy danych zdefiniowanej przez to polecenie, *`typ`* oznacza typ bazy danych, a *`argumenty`* definiują używaną bazę danych. Argumenty mają format:

```
znaczniki opis
```

gdzie *`znaczniki`* definiują opcje używane przez bazę danych, a *`opis`* identyfikuje używaną bazę danych. W większości przypadków *`opis`* zawiera ścieżkę do zewnętrznej bazy danych — albo do lokalnej bazy danych, albo do mapy dostępnej w serwerze bazodanowym. Jednakże w przypadku wyrażeń regularnych opis jest definicją wyrażenia regularnego, z którym porównywane są dane wejściowe. W naszym przykładzie polecenie `K` wygląda następująco:

```
Kcheckaddress regex -a@MATCH
   ^([0-9]+<@(aol|msn)\.com|[0-9][^<]*<@juno\.com)\.?>
```

W tym przykładzie:

- `K` jest poleceniem.
- `checkaddress` jest nazwą wewnętrzną.
- `regex` jest typem.
- `-a@MATCH` jest znacznikiem mówiącym sendmailowi, że po znalezieniu dopasowania ma zostać zwrócona wartość `@MATCH`.
- `^([0-9]+<@(aol|msn)\.com|[0-9][^<]*<@juno\.com)\.?>` jest wyrażeniem regularnym. Jest to standardowe wyrażenie regularne, którego można użyć z narzędziami typu *egrep* i *awk*. To przykładowe wyrażenie regularne dopasowuje adresy e-mail z *aol.com*, *msn.com* i *juno.com* zawierające numeryczne nazwy użytkowników.

Polecenie `K` definiuje wyrażenie regularne, lecz do jego użycia potrzebna będzie reguła przekształceń. Makro `LOCAL_RULESETS` posłużyło do wstawienia niestandardowego zestawu reguł do pliku *sendmail.cf*. Podstawą przykładowego zestawu reguł `Local_check_mail` są trzy polecenia `R`:

```
R$*                             $: $>Parse0 $>3 $1
R$+                             $: $(checkaddress $1 $)
R@MATCH                         $#error $: "553 Header error"
```

Adres przekazany do zestawu reguł `Local_check_mail` jest najpierw przetwarzany przez zestaw reguł nr 3 (inaczej zwany `canonify`), a wyniki tego procesu są przesyłane przez zestaw reguł `Parse0`. Proszę zwrócić uwagę, że oba zestawy reguł są wywoływane przez pierwsze polecenie przekształcenia. To przetwarzanie sprowadza adres do postaci

kanonicznej. Następnie adres jest przyrównywany do wzorca z wyrażenia regularnego `checkaddress` przez drugą regułę przekształceń. Jeśli adres pasuje do wyrażenia regularnego, to zostaje zastąpiony łańcuchem `@MATCH`. Trzecia reguła przekształcenia sprawdza, czy przestrzeń robocza zawiera ten łańcuch. Jeśli tak, zwracany jest błąd nagłówka.

Kilka testów pokaże, jak działają wyrażenia regularne i zestaw reguł:

```
# sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> Local_check_mail 123@aol.com
Local_check_mail   input: 123 @ aol . com
canonify           input: 123 @ aol . com
Canonify2          input: 123 < @ aol . com >
Canonify2        returns: 123 < @ aol . com . >
canonify         returns: 123 < @ aol . com . >
Parse0             input: 123 < @ aol . com . >
Parse0           returns: 123 < @ aol . com . >
Local_check_mail returns: $# error $: "553 Header error"
> Local_check_mail win@aol.com
Local_check_mail   input: win @ aol . com
canonify           input: win @ aol . com
Canonify2          input: win < @ aol . com >
Canonify2        returns: win < @ aol . com . >
canonify         returns: win < @ aol . com . >
Parse0             input: win < @ aol . com . >
Parse0           returns: win < @ aol . com . >
Local_check_mail returns: win < @ aol . com . >
 > /quit
```

Pierwszy test przekazuje adres *123@aol.com* do zestawu reguł `Local_check_mail`. Ten adres powinien pasować do wyrażenia regularnego `checkaddress`. Z błędu zwróconego przez zestaw reguł `Local_check_mail` wynika, że tak jest. Drugi test pokazuje, że poprawny adres z *aol.com* nie generuje tego błędu.

Powyższy przykład, wzięty z pliku *knecht.mc*, nie jest zaleceniem, że powinniśmy odfiltrowywać liczbowe adresy z *aol.com*. Jest to tylko przykład sposobów definiowania i używania wyrażeń regularnych. Makro `LOCAL_CONFIG`, makro `LOCAL_RULESETS` i składnia polecenia `K` są takie same dla tworzonych przez nas wyrażeń regularnych i własnych zestawów reguł jak w tym prostym przykładzie.

Zobacz również

Receptura 6.9 zawiera dodatkowe informacje, które powinny pomóc w zrozumieniu niniejszej receptury. Rozdział 1. opisuje proces kompilacji sendmaila. Książka *sendmail* omawia makro `LOCAL_CONFIG` w podpunkcie 4.3.3.1,[12] `LOCAL_RULESETS` w 4.3.3.5[13] i typ mapy `regex` w punkcie 23.7.20. Książka wydawnictwa O'Reilly *Mastering Regular Expressions*[14] zawiera

[12] Wydanie polskie: *sendmail*, Helion, 2001, punkt 19.6.30.

[13] Wydanie polskie: *sendmail*, Helion, 2001, punkt 19.6.36.

[14] Wydanie polskie: *Wyrażenia regularne*, Helion, 2001.

szczegółowy opis wyrażeń regularnych. Dodatkowe informacje o poleceniach *sendmail.cf* zawierają książki *TCP/IP Network Administration, Third Edition* Craiga Hunta (O'Reilly) i *Linux Sendmail Administration* Craiga Hunta (Sybex).

6.11. Identyfikowanie użytkowników lokalnych sprawiających problemy

Problem

Spamerzy ukrywają swoją prawdziwą tożsamość. Chcemy dostarczać tyle informacji, ile tylko możliwe, aby wyśledzić spamerów korzystających z naszego systemu.

Rozwiązanie

Uruchom usługę `auth` (*identd*), aby dostarczać informacji o koncie, których nie można ukryć maskaradą i innymi technikami.

Analiza

Protokół IDENT został zdefiniowany w RFC 1413, *Identification Protocol*. Protokół ten udostępnia środki identyfikacji użytkownika, który zainicjował połączenie sieciowe. Demon *identd* implementuje protokół IDENT w systemach uniksowych. Uruchomienie *identd* pozwala dostarczać dodatkowych informacji administratorom zdalnych systemów, co pozwoli im śledzić źródła problemów. W kontekście sendmaila informacje te pozwalają dotrzeć do spamera, jeśli prowadzi swój biznes z naszego systemu. Załóżmy na przykład, że użytkownik w systemie używającym *identd* spróbuje sfałszować swoją tożsamość, wydając polecenie SMTP `EHLO` z użyciem fałszywej nazwy hosta:

```
ehlo www.ora.com
250-rodent.wrotethebook.com
Hello IDENT:r+9Gemj2wip8fAJDU8kDZlyUiReTZjYc@chef.wrotethebook.com
[192.168.0.8], pleased to meet you
```

Gdy zdalny system, w tym przypadku *rodent.wrotethebook.com*, odpowie na `EHLO`, to zignoruje nazwę hosta *www.ora.com*, a zamiast tego przywita się z hostem, którego znajdzie pod adresem połączenia 192.168.0.8, podając nazwę hosta skojarzoną z tym adresem (w tym przypadku *chef.wrotethebook.com*) oraz informacje identyfikujące, których dostarcza usługa *identd* uruchomiona w hoście *chef*. Informacje te są przekazywane w wiadomości w nagłówku `Received:`, jak widać poniżej:

```
Received: from www.ora.com
  (IDENT:r+9Gemj2wip8fAJDU8kDZlyUiReTZjYc@chef [192.168.0.8])
   by rodent.wrotethebook.com (8.12.9/8.12.9) with ESMTP id gB4N6T301540
   for <craig@rodent.wrotethebook.com>; Wed, 4 Dec 2002 18:06:40 -0500
```

Informacje udostępnione przez usługę *identd* działającą w hoście *chef* pod adresem 192.168.0.8 identyfikują użytkownika, który wysłał wiadomość. Łańcuch dostarczany przez *identd*, w naszym przykładzie `r+9Gemj2wip8fAJDU8kDZlyUiReTZjYc`, nie jest zwykłą nazwą użytkownika. W tym przypadku informacje *identd* są szyfrowane.

identd monitoruje port 113. Gdy demon *identd* jest uruchomiony, zdalny serwer może zażądać informacji o wszelkich połączeniach TCP z naszego serwera do serwera zdalnego, wysyłając parę portów, źródłowy i docelowy, do serwera identyfikacji. Wówczas *identd* odpowiada, odsyłając albo zażądane informacje związane z połączeniem, albo błąd. Informacje te pozwalają zdalnym serwerom poczty umieścić prawdziwą nazwę użytkownika w nagłówku `Received:` przychodzącej wiadomości. Użytkownicy nadużywający systemów pocztowych nie lubią ujawniać swoich prawdziwych nazw użytkowników. Przesłanie tej nazwy ofiarom utrudnia spamerom pozostanie w biznesie. Niestety, wiele zapór sieciowych blokuje port 113, ponieważ osoby odpowiedzialne za bezpieczeństwo obawiają się, że usługa *identd* ujawnia zbyt wiele informacji. Te obawy nie mają podstaw, gdy informacje *identd* są szyfrowane, jak w tym przykładzie. Jednakże wielu administratorów woli dmuchać na zimne i blokować port. Jeśli port 113 jest zablokowany na zaporze sieciowej, nie będziemy przypuszczalnie mogli skorzystać z *identd*.

Protokół IDENT jest znany również pod nazwą usługi `auth`, jak widać z poniższego wyszukiwania grep w */etc/services*:

```
$ grep ^auth /etc/services
auth    113/tcp    ident   # User Verification
```

Większość administratorów sendmaila woli nazywać tę usługę *ident* lub *identd*, aby uniknąć pomylenia jej ze słowem kluczowym ESMTP `AUTH`, które zostało omówione w rozdziale 7. Oprócz tego *identd* nie jest narzędziem służącym do uwierzytelniania, lecz do inspekcji. Prawdziwi spamerzy, którzy mają kontrolę nad własnymi systemami, mogą w odpowiedzi *identd* umieścić, co tylko chcą, więc odpowiedź ta nie może wiarygodnie posłużyć do uwierzytelniania. My używamy *identd*, aby dysponować dodatkowymi informacjami pozwalającymi wyśledzić użytkowników nadużywających naszego systemu.

Wiele systemów uniksowych uruchamia *identd* na żądanie z `inetd` lub `xinetd`. Poniższy wiersz dodany do *inetd.conf* zaimplementuje recepturę 6.11 w systemie używającym *inetd* oraz na żądanie usługi *identd*:

```
auth stream tcp nowait nobody /usr/sbin/in.identd in.identd -t120
```

To oczywiście tylko przykład. Ścieżkę do programu i argumenty wiersza polecenia będziemy musieli dopasować do naszych potrzeb. Czytelnik powinien też sprawdzić w dokumentacji man *identd* opcje uniemożliwiające użytkownikowi wyłączenie *identd*.

Niektóre systemy uniksowe, jak np. nasz przykładowy system Red Hat Linux, uruchamiają *identd* podczas startu systemu. Polecenie *chkconfig* dodaje *identd* do procesu startu systemu, a polecenie *service* uruchamia *identd* natychmiast:

```
# chkconfig --list identd
identd          0:off   1:off   2:off   3:off   4:off   5:off   6:off
# chkconfig --level 35 identd on
```

```
# chkconfig --list identd
identd          0:off   1:off   2:off   3:on    4:off   5:on    6:off
# service identd start
Generating ident key:                                      [  OK  ]
Starting identd:                                           [  OK  ]
```

Pierwsze polecenie *chkconfig* pokazuje, że *identd* nie jest zawarty w procesie uruchamiania przykładowego systemu Linux. Drugie polecenie *chkconfig* dodaje *identd* do procesu startu dla poziomów działania 3. i 5., związanych z sieciowym wieloużytkownikowym działaniem w większości systemów Linux. Ostatnie polecenie *chkconfig* pokazuje skutki tej zmiany. Oczywiście nie ma powodu, by restartować system tylko w celu uruchomienia *identd*, więc użyliśmy polecenia `service`, aby natychmiast uruchomić usługę identyfikacji.

To było pierwsze uruchomienie *identd* w tym serwerze. Proszę zwrócić uwagę na pierwszy wiersz wyjścia polecenia *service*, który mówi, że dla *identd* jest generowany klucz. Klucz ten służy do szyfrowania informacji wysyłanych do zdalnego systemu. Administratorzy zdalnego systemu nie mogą odszyfrować tych informacji, ponieważ nie dysponują kluczem. Jeśli któryś z nich podejrzewa problem, musi wysłać do nas zaszyfrowany łańcuch, który odszyfrujemy, używając pliku */etc/identd.key* i polecenia *idecrypt*. Informacje pochodzące z serwera *identd* są dla bezpieczeństwa szyfrowane. Osoby odpowiedzialne za bezpieczeństwo zwykle nie lubią *identd*, ponieważ ta usługa ujawnia informacje o naszych systemach i użytkownikach, które mogą nadużyć spamerzy lub intruzi. Szyfrowanie odpowiedzi *identd* pozwala uruchomić demona identyfikacji bez poważnych zagrożeń.

Na przykład, nagłówek `Received:` pokazany wcześniej w tym punkcie wyświetla 32-znakowy zaszyfrowany łańcuch zakodowany w BASE64: `r+9Gemj2wip8fAJDU8kDZlyUi➥ReTZjYc`, który zdalny system otrzymał z naszego przykładowego systemu w odpowiedzi na zapytanie IDENT. Nie ujawnia to żadnych informacji, które mógłby wykorzystać spamer lub intruz, lecz możemy użyć takiego łańcucha do otrzymania informacji o lokalnym użytkowniku, który wysłał wiadomość. Gdy administrator zdalnego systemu skontaktuje się z nami, zgłaszając problem, weźmiemy ten łańcuch, wpiszemy w nawiasy prostokątne i odszyfrujemy poleceniem `idecrypt`:

```
# idecrypt
[r+9Gemj2wip8fAJDU8kDZlyUiReTZjYc]
Wed Dec  4 17:00:24 2002 500 192.168.0.8 1029 192.168.0.3 25
Ctrl-D
```

Odszyfrowany łańcuch podaje:

- Datę i godzinę połączenia.
- UID użytkownika, który zainicjował połączenie (w naszym przykładzie 500).
- Źródłowy adres IP połączenia (192.168.0.8).
- Port źródłowy (1029).
- Docelowy adres IP (192.168.0.3).
- Docelowy port połączenia (25 czyli port SMTP).

Wszystkie te informacje są przydatne do śledzenia, kto nadużywa naszego systemu. Jednakże *identd* ma poważne ograniczenia. Zwraca użyteczny UID tylko wtedy, gdy użytkownik faktycznie zaloguje się do systemu. Jeśli system będzie nadużywany w inny sposób, udostępnione informacje niekoniecznie będą przydatne. Oprócz tego używanie *identd* zwiększa obciążenie systemu przy przetwarzaniu każdej wiadomości. Podobnie jak wszystko inne, *identd* ma swoje plusy i minusy.

W naszym przykładowym systemie Red Hat Linux usługa `identd` jest wstępnie skonfigurowana w pliku */etc/identd.conf*. Oprócz tego usługa `identd` może być konfigurowana z poziomu wiersza poleceń za pomocą opcji wiersza polecenia. W systemie Red Hat Linux opcje wiersza polecenia `identd` używane podczas startu systemu są definiowane jako wartości dla zmiennej `IDENTOPTS` w pliku */etc/sysconfig/identd*. Szczegóły poleceń konfiguracji *identd.conf* i opcji wiersza polecenia zawiera dokumentacja man `identd`.

Zobacz również

Dla *identd*, *inetd.conf*, *chkconfig* i *service* dostępna jest dokumentacja systemowa (strony man).

6.12. Używanie programów MILTER

Problem

Zainstalowaliśmy zewnętrzne narzędzie filtrujące pocztę zgodne z *Sendmail Mail Filter API*. Teraz należy skonfigurować sendmail tak, by korzystał z tego zewnętrznego filtru poczty.

Rozwiązanie

Uruchom polecenie `sendmail -d0.1`. Wiersz wyjścia „Compiled with:" polecenia powinien zawierać MILTER. Jeśli nie, dodaj obsługę `MILTER` przez dodanie do pliku *site.config.m4* następującego wiersza:

```
APPENDDEF(`confENVDEF', `-DMILTER')
```

Zrekompiluj sendmail jak w rozdziale 1., w recepturach od 1.3 do 1.7.

Dodaj do konfiguracji sendmaila makro `INPUT_MAIL_FILTER`, które identyfikuje zewnętrzny filtr poczty. Makro `INPUT_MAIL_FILTER` powinno definiować przynajmniej wewnętrzną nazwę filtru i specyfikację gniazda wymaganego przez filtr. Sprawdź w dokumentacji programu MILTER zalecane dla niego ustawienia. Na przykład poniższy wpis do konfiguracji sendmaila pozwala korzystać z programu MILTER *vbsfilter* dostępnego pod adresem *http://aeschi.ch.eu.org/milter/*:

```
INPUT_MAIL_FILTER(`sample', `S=local:/var/run/vbsfilter.sock')
```

Skompiluj plik *sendmail.cf*, skopiuj do */etc/mail* i uruchom ponownie sendmail, jak w recepturze 1.8.

Analiza

Aby sendmail mógł używać programu zewnętrznego, program ten musimy znaleźć, pobrać, zainstalować i poprawnie skonfigurować. Wstęp do niniejszego rozdziału zawiera adresy kilku witryn WWW, od których możemy zacząć poszukiwania przydatnych programów MILTER. Z uwagi na złożoność filtrowania spamu i skanowania w poszukiwaniu wirusów oraz z powodu nieustającej zmienności zagrożeń, skonfigurowanie programu MILTER może być o wiele trudniejsze niż skonfigurowanie sendmaila do współpracy z tym programem. Radzę dokładnie przeczytać dokumentację MILTER, aby dokładnie zrozumieć, jakie argumenty w makrze `INPUT_MAIL_FILTER` są niezbędne, by sendmail mógł współpracować z filtrem zewnętrznym.

Przykładowa konfiguracja sendmaila z rozwiązania zawiera makro `INPUT_MAIL_FILTER` zalecane przez dokumentację filtru poczty *vbsfilter*. Program *vbsfilter* identyfikuje różne niebezpieczne załączniki i zmienia rozszerzenia ich nazw na *.txt*, aby nie były automatycznie wykonywane przez system użytkownika końcowego. Gniazdo zdefiniowane w makrze `INPUT_MAIL_FILTER` musi być zgodne z gniazdem używanym przez MILTER. W przypadku programu *vbsfilter* jest to osiągnięte przez uruchomienie MILTER z argumentem `-p`, który informuje *vbsfilter*, którego gniazda program ma użyć. W przypadku makra `INPUT_MAIL_FILTER` z rozwiązania program *vbsfilter* będzie uruchamiany następującym poleceniem:

```
# vbsfilter -p S=local:/var/run/vbsfilter.sock
```

vbsfilter jest dobrym programem na początek, ponieważ nie wymaga zbyt wiele konfigurowania, podobnie jak przykładowy MILTER znajdujący się w pliku *libmilter/README* zawartym w dystrybucji sendmaila. Przykładowy filtr poczty nie jest specjalnie przydatny do jakiegoś filtrowania, lecz jest łatwy do przetestowania i może pomóc w ustaleniu, czy sendmail jest poprawnie skonfigurowany i potrafi komunikować się z filtrem zewnętrznym. Wiele filtrów jest tak złożonych, że uruchamianie programu MILTER przy jednoczesnym uruchamianiu sendmaila może być przytłaczające. Przed połączeniem ze skomplikowanym filtrem warto uruchomić sendmail z prostym filtrem.

Zobacz również

Wstęp do niniejszego rozdziału zawiera dodatkowe informacje o programach MILTER. Książka *sendmail* omawia MILTER w podrozdziale 7.6. Temat ten jest również omówiony w pliku *libmilter/README* oraz w plikach w *libmilter/docs*.

6.13. Omijanie kontroli spamu

Problem

Nasz sendmail jest skonfigurowany do blokowania przychodzącego spamu, a chcemy przepuszczać reklamy adresowane do konkretnych odbiorców.

Rozwiązanie

Dodaj do pliku tekstowego */etc/mail/access* wpis dla każdego odbiorcy, który będzie mógł odbierać spam. Pole klucza we wpisie utwórz ze słowa kluczowego `Spam:` i adresu odbiorcy, a jako wartość zwracaną wpisz słowo kluczowe `FRIEND`. Uruchom *makemap*, aby zbudować z pliku tekstowego bazę danych typu hash.

Utwórz konfigurację sendmaila zawierającą funkcję *access_db* i funkcję *delay_checks* z opcjonalnym argumentem `friend`. Upewnij się, że makro `FEATURE` *access_db* znajduje się w konfiguracji przed makrem `FEATURE` *delay_checks*. Do konfiguracji sendmaila powinny zostać dodane poniższe wpisy:

```
dnl Użyj bazy danych access
FEATURE(`access_db')
dnl Przed odrzuceniem wiadomości sprawdź, kto lubi spam
FEATURE(`delay_checks', `friend')
```

Skompiluj plik *sendmail.cf*, skopiuj nowy *sendmail.cf* do */etc/mail* i uruchom ponownie sendmail, jak w recepturze 1.8.

Analiza

Komuś w naszym systemie — użytkownikowi *postmaster*, ekspertowi od zabezpieczeń, programiście piszącemu filtry poczty — może być potrzebne odbieranie spamu domyślnie blokowanego przez sendmail. Funkcja *delay_checks* pozwala na to, zmieniając kolejność stosowania kontroli spamu. Funkcja *delay_checks* pozwala na sprawdzenie adresu odbiorcy koperty przed adresem nadawcy koperty lub adresem połączenia, co z kolei umożliwia ominięcie przez wiadomość zaadresowaną do konkretnego nadawcy tych dwóch kontroli. Aby użyć *delay_checks* w ten sposób, musimy wywołać funkcję z argumentem `Friend`, jak w pokazanym powyżej rozwiązaniu.

Konkretni odbiorcy, którzy będą mogli odbierać spam, są zdefiniowani w bazie danych *access* z użyciem znacznika `Spam:` i wartości zwracanej `FRIEND`. Oto przykład bazy danych *access*:

```
Connect:example.com                 REJECT
Spam:uce@wrotethebook.com           FRIEND
Spam:clark+junk@wrotethebook.com    FRIEND
```

W przypadku konfiguracji sendmaila opisanej w rozwiązaniu ta baza danych *access* odrzuca pocztę z *example.com*, o ile nie jest zaadresowana do *uce@wrotethebook.com* lub *clark+junk@wrotethebook.com*.

Zobacz również

Receptura 6.14 zawiera podobny przykład, choć dotyczący przeciwnego rozwiązania. Rozdział 3. i wstęp do niniejszego rozdziału zawierają dodatkowe informacje o bazie danych *access*. Książka *sendmail* omawia bazę danych *access* w podrozdziale 7.5 i funkcję *delay_checks* w punkcie 7.5.6. Temat ten jest też omawiany w sekcji *Delay all checks* pliku *cf/README*.

6.14. Włączenie kontroli spamu dla poszczególnych użytkowników

Problem

Należy utworzyć konfigurację sendmaila, która stosuje kontrole tylko wtedy, gdy poczta jest adresowana do wybranych odbiorców.

Rozwiązanie

Dodaj do pliku tekstowego */etc/mail/access* wpis dla każdego odbiorcy, którego poczta będzie musiała przed doręczeniem przejść wszystkie kontrole. Pole klucza we wpisie utwórz ze słowa kluczowego `Spam:` i adresu odbiorcy, a jako wartość zwracaną wpisz słowo kluczowe `HATER`. Uruchom *makemap*, aby zbudować z pliku tekstowego bazę danych typu hash.

Utwórz konfigurację sendmaila zawierającą funkcję *access_db* i funkcję *delay_checks* z opcjonalnym argumentem `hater`. Upewnij się, że makro `FEATURE` *access_db* znajduje się w konfiguracji przed makrem `FEATURE` *delay_checks*. Do konfiguracji sendmaila powinny zostać dodane takie wpisy:

```
dnl Użyj bazy danych access
FEATURE(`access_db')
dnl Zastosuj wszystkie kontrole do poczty użytkowników nienawidzących spamu
FEATURE(`delay_checks', `hater')
```

Skompiluj plik *sendmail.cf*, skopiuj nowy *sendmail.cf* do */etc/mail* i uruchom ponownie sendmail, jak w recepturze 1.8.

Analiza

Funkcja *delay_checks* zmienia kolejność, w której kontrole są stosowane, zaczynając od sprawdzenia adresu odbiorcy koperty. Gdy funkcja *delay_checks* jest wywoływana z argumentem `hater`, jak w rozwiązaniu, adres odbiorcy koperty musi znajdować się w bazie danych *access* i musi zwracać wartość `HATER`, aby adres nadawcy koperty lub adres połączenia mógł zostać sprawdzony. Jeśli adres nadawcy koperty nie zostanie znaleziony w bazie danych *access*, lub nie zwróci wartości `HATER`, zostaną ominięte kontrola adresu nadawcy koperty przez `check_mail` i kontrola adresu połączenia przez `check_relay`.

Jeżeli chcemy zastosować kontrole `check_mail` i `check_relay` do poczty konkretnego odbiorcy, odbiorca ten musi być zdefiniowany w bazie danych *access* z użyciem znacznika `Spam:` i wartości zwracanej `HATER`. Oto przykładowe wpisy:

```
Connect:example.com                 REJECT
Spam:jay@wrotethebook.com           HATER
Spam:alana@wrotethebook.com         HATER
```

Przy użyciu konfiguracji sendmaila z niniejszej receptury baza danych *access* będzie odrzucać wiadomości z *example.com* zaadresowane do użytkowników *jay@wrotethebook.com* i *alana@wrotethebook.com*. Pozostali użytkownicy będą mogli odbierać pocztę z *example.com*.

Zobacz również

Receptura 6.13 zawiera przykład podobny, choć dotyczący przeciwnego problemu. Rozdział 3. i wstęp do niniejszego rozdziału zawierają dodatkowe informacje o bazie danych *access*. Książka *sendmail* omawia bazę danych *access* w podrozdziale 7.5 i funkcję *delay_checks* w punkcie 7.5.6. Temat ten jest też omawiany w sekcji *Delay all checks* pliku *cf/README*.