

**WYŻSZA SZKOŁA INFORMATYKI STOSOWANEJ
I ZARZĄDZANIA**

pod auspicjami Polskiej Akademii Nauk



WYDZIAŁ INFORMATYKI

STUDIA I STOPNIA
(INŻYNIERSKIE)



PRACA DYPLOMOWA

Maciej Przeor

PROJEKT I REALIZACJA SERWERA LINUKSOWEGO NA POTRZEBY MAŁEJ SIECI LOKALNEJ

Praca wykonana pod kierunkiem:

dr Piotr Koperski

WARSZAWA, 2009 r.

Autor: **Maciej Przeor**

Tytuł: **PROJEKT I REALIZACJA SERWERA LINUKSOWEGO NA
POTRZEBY MAŁEJ SIECI LOKALNEJ**

Rok akademicki: **2008/2009**

Dziekan Wydziału: **dr inż. Jarosław Sikorski**

Specjalność: **Sieci komputerowe**

Opiekun pracy: **dr Piotr Koperski**

**Kwalifikuję do złożenia jako dyplomową pracę inżynierską
na Wydziale Informatyki WSISiZ.**

Ocena pracy (słownie):

(skala ocen: bardzo dobra, dobra i pół, dobra, dostateczna i pół, dostateczna)

.....
(data)

.....
(podpis opiekuna pracy)

Pracę złożono w egzemplarzach dnia

(pieczęć wydziału)

.....
(podpis kierownika dziekanatu)

Tytuł naukowy, imię i nazwisko recenzenta pracy:

.....

Ocena pracy (słownie):

(skala ocen: bardzo dobra, dobra i pół, dobra, dostateczna i pół, dostateczna)

.....
(data)

.....
(podpis kierownika dziekanatu)

Dopuszczam pracę do obrony.

.....
(data)

.....
(podpis dziekana)

Autor: **Maciej Przeor**

Tytuł: **PROJEKT I REALIZACJA SERWERA LINUXOWEGO NA POTRZEBY
MAŁEJ SIECI LOKALNEJ**

Rok akademicki: **2008/2009**

Dziekan Wydziału: **dr inż. Jarosław Sikorski**

Specjalność: **Sieci komputerowe**

Opiekun pracy: **dr. Piotr Koperski**

OŚWIADCZENIE AUTORA PRACY DYPLOMOWEJ

Świadom(a) odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami prawa, w szczególności z ustawą z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz. U. z 1994 r. Nr 24, poz. 83 – tekst pierwotny i Dz. U. z 2000 r. Nr 80, poz. 904 – tekst jednolity, z późniejszymi zmianami).

Oświadczam, że wszystkie narzędzia informatyczne zastosowane do wykonania niniejszej pracy wykorzystałem(am) zgodnie z obowiązującymi przepisami prawa w zakresie ochrony własności intelektualnej i przemysłowej.

Oświadczam, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w szkole wyższej.

Jednocześnie stwierdzam, że niniejszy tekst pracy dyplomowej jest identyczny z załączoną wersją elektroniczną.

.....
Data

.....
Podpis autora pracy

Wstęp	6
1. Linuks jako system operacyjny	7
1.1 Jądro Linuksa	7
1.1.1 Kompilacja jądra	8
1.2 Program startowy (bootloader)	9
1.3 Funkcje systemu	9
1.3.1 Filtrowanie pakietów	10
1.3.2 Wymiana plików w sieci	10
1.3.3 Wirtualna sieć prywatna	11
1.3.4 Zarządzanie i monitoring	11
1.4 Wykorzystywany sprzęt	12
1.5 Porównanie i wybór dystrybucji	14
1.5.1 Gentoo	14
1.5.2 Debian GNU/Linux	15
1.5.3 Vyatta	15
1.5.4 Trustix Secure Linux	16
1.5.5 Fedora	17
1.5.6 Core GNU/Linux	17
1.6 Wybór systemu plików	18
1.7 Założenia projektu	18
2. Realizacja serwera	19
2.1 Instalacja Core 2.0	20
2.2 Konfiguracja GRUB'a	21
2.3 Kompilacja jądra	23
2.4 Mcron	27
2.5 Konfiguracja sieci	28
2.5.1 Klient DHCP	28
2.5.2 Ręczna konfiguracja protokołu TCP/IP	31
2.5.3 Serwer DHCP	32
2.6 OpenSSH	33
2.7 Iptables	35
2.8 Lighttpd + PHP	40
2.9 Samba	42
2.10 Poptop – PPTP Server	44

2.11	Zarządzanie usługami	46
3.	Zarządzanie serwerem i testy systemu	48
3.1	Zarządzanie serwerem	48
3.2	Testy systemu.....	60
3.2.1	Nmap – skanowanie portów	60
3.2.2	Ilość połączeń TCP oraz UDP	63
3.2.3	Czas uruchamiania systemu	63
3.2.4	Podsumowanie testów.....	65
	Podsumowanie i wnioski.....	66
	Bibliografia	67

Wstęp

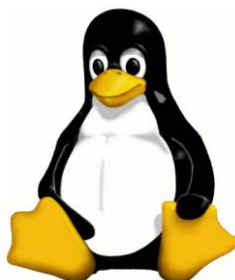
Wraz z dużym rozwojem Internetu na świecie zwiększa się liczba instalowanych routerów i serwerów pełniących funkcję routera. Coraz częściej w domach i prawie w każdej firmie montowane są urządzenia pozwalające rozdzielać Internet między kilka komputerów. Początkowo sprzęt ten pełnił tylko podstawowe funkcje, obecnie nawet najprostsze routery wyposażone są w moduł radiowy, zarządzanie przez przeglądarkę czy przekierowywanie portów. Zróżnicowanie routerów na rynku jest ogromne jednak tanie produkty są mało wydajne i posiadają ograniczone możliwości a wydajne i funkcjonalne swoją ceną znacznie przekraczają budżet przeciętnego gospodarstwa domowego.

Celem pracy jest zaprojektowanie i realizacja wydajnego i funkcjonalnego serwera na potrzeby małych sieci lokalnych, którego koszty będą zbliżone do najtańszych routerów. W pracy omówiono dokładnie wszystkie kroki jakie zostały podjęte aby zrealizować taki system.

Praca składa się z trzech rozdziałów. W pierwszym rozdziale przedstawiono ogólne informacje o systemie Linuks oraz wymieniono funkcję jaki ma pełnić projektowany system. W drugim rozdziale opisano proces instalacji i konfiguracji oprogramowania. Rozdział trzeci omawia sposób zarządzania serwerem oraz przedstawia testów wydajności i bezpieczeństwa serwera.

1. Linuks jako system operacyjny

Termin Linuks formalnie określa jądro systemu, jednak większość z nas używa tego słowa opisując całą dystrybucję a więc traktuje go jako darmowy system operacyjny, który steruje i nadzoruje pracę komputera. Dystrybucja Linuksa (GNU/Linux) składa się z jądra oraz z zestawu oprogramowania niezbędnego do działania.



Rys. 1.1. logo Linuksa (źródło: <http://thecamels.org>)

1.1 Jądro Linuksa

Jądro jest jedną z części systemu operacyjnego, jego zadaniem jest obsługa systemu plików, zarządzanie pamięcią, przydzielanie czasu procesora oraz urządzeń zewnętrznych. Jądro systemu jest programem napisanym w języku C z drobnymi wstawkami assemblerowymi. Dostępne jest w postaci kodu źródłowego lub skompilowanych pakietów binarnych. W jądrze znajdują się programy odpowiedzialne za działanie komputera oraz sterowniki dla większości urządzeń dostępnych na rynku. Jądro ładowane jest do pamięci operacyjnej przez bootloader, posiada ono wiele niezbędnych dla systemu procedur oraz dostarcza środowisko dla aplikacji działających w systemie. Jądro w systemach Linuksowych odgrywa rolę pośrednika między programami a sprzętem.

Pierwotna wersja jądra Linuksa została stworzona w 1991 roku na podstawie jądra Minixa, systemu unixowego o niewielkich wymaganiach sprzętowych. Od tamtego czasu regularnie udostępniane są nowe wersje kodu źródłowego. Można wyróżnić dwie wersje jądra: stabilną oraz rozwojową. Wersja stabilna przeznaczona jest dla użytkowników ceniących sobie stabilność oraz niezawodność systemu. Wersja rozwojowa zawiera nowe, nie do końca przetestowane funkcje oraz sterowniki dla najnowszych urządzeń mogące zakłócać prawidłową pracę systemu

Twórcy jądra oprócz całych źródeł programu udostępniają również łatki, zawierające tylko ostatnie poprawki z danej serii, dzięki czemu nie trzeba pobierać całych źródeł po każdej aktualizacji.

Od późniejszej wersji 1.3 jądro umożliwia instalowanie opcji jako moduły, dzięki czemu nie są one bezpośrednio wkompilowane w jądro. W pamięci operacyjnej znajdują się tylko wybrane opcje, niezbędne do działania systemu, takie jak obsługa systemu plików czy dysku twardego. Gdy zajdzie potrzeba użycia danego modułu zostaje on załadowany dynamicznie do pamięci bez przerywania pracy systemu. Po zakończeniu operacji na danym urządzeniu moduł można wykasować z pamięci operacyjnej.

1.1.1 Kompilacja jądra

Standardowe jądro dostarczane wraz z większością dystrybucji obsługuje znaczną część konfiguracji sprzętowych i ponowna kompilacja jądra na stacji roboczej nie jest najczęściej potrzebna. Sytuacja ma się inaczej w przypadku serwerów, które zoptymalizowane są pod kątem pełnienia konkretnych funkcji, tutaj kompilacja jest zazwyczaj zalecana:

- w celu uaktualnienia oprogramowania i załatwienia dziur
- gdy korzysta się ze specjalistycznego sprzętu
- w przypadku gdy chce się korzystać z wersji rozwojowej jądra
- w celu dodania nowych funkcji, które są wyłączone w standardowym jądrze
- w przypadku gdy chce się przyspieszyć pracę systemu optymalizując rozmiar jądra przez usunięcie niepotrzebnych sterowników

Dzięki zastosowaniu modularności jądro staje się mniejsze a cały system szybszy i wydajniejszy gdyż zbędne funkcje nie zajmują cennych zasobów komputera. Dodatkowo poprzez zastosowanie modułów można zaoszczędzić wiele pracy przy dodawaniu nowego sprzętu ponieważ nie trzeba ponownie kompilować całego jądra. Warto dodać że moduły są kompilowane niezależnie od jądra.

Konfiguracja jądra jest najtrudniejszym i najbardziej czasochłonnym etapem kompilacji. Wymaga od użytkownika wiedzy o Linuksie, dobrej znajomości sprzętu oraz ściśle sprecyzowanych funkcji jakie będzie pełniła dana maszyna. Istnieją trzy sposoby konfiguracji jądra:

- *make config* – najstarsza metoda działająca w trybie tekstowym, użytkownik odpowiada pojedynczo na pytania dotyczące każdej opcji w jądrze
- *make menuconfig* – metoda pozwalająca wybrać odpowiednie funkcje w jądrze za pomocą tekstowego menu
- *make xconfig* – najwygodniejsza metoda, działająca tylko w trybie graficznym

Ważne jest aby zaznaczać tylko niezbędne opcje gdyż zwiększy to szybkość działania jądra, zmniejszy jego rozmiar oraz także skróci czas kompilacji.

1.2 Program startowy (bootloader)

Bootloader to niewielki program uruchamiany jako pierwszy tuż po wykonaniu się BIOSu. Składa się on najczęściej z kodu wykonywalnego, tak zwanego bootstrap oraz z pliku konfiguracyjnego. Bootstrap może być umieszczony na dyskietce, nośniku optycznym lub pamięci flash, choć najczęściej znajduje się w głównym rekordzie startowym dysku. Plik konfiguracyjny przechowuje wszystkie potrzebne ustawienia z których korzysta bootstrap. Podstawowym zadaniem programu rozruchowego jest załadowanie systemu do pamięci operacyjnej, jednak większość z aktualnie stosowanych programów pełni również funkcje menadżera uruchamiania dzięki czemu możemy wybrać, który system ma się uruchomić.

Lilo to jedno z najstarszych i najpopularniejszych narzędzi do uruchamiania systemu Linuks. Posiada wiele opcji a jednocześnie jest bardzo proste w konfiguracji. Nie jest zależne od żadnego systemu plików, potrafi załadować jądro systemu zarówno z dyskietki jak i z dysku twardego. W łatwy sposób może być użyte do uruchomienia większości systemów operacyjnych. Lilo umieszcza się w głównym sektorze startowym dysku twardego lub w sektorze startowym aktywnej partycji. Konfiguracja programu przechowywana jest najczęściej w pliku */etc/lilo.conf*.

GRUB^[1] to kolejny bootloader, zdobył on sporą popularność dzięki swojej elastyczności oraz dużym możliwościom. W przeciwieństwie do Lilo potrafi odczytać bardzo wiele systemów plików umożliwiając tym samym przechowywanie pliku konfiguracyjnego na partycji a nie w sektorze startowym. Do jednej z największych zalet GRUBa należy możliwość edycji konfiguracji z poziomu działającego bootloadera.

GRUB w odróżnieniu od Lilo nie używa nazw dysków opierających się na urządzeniach widniejących w katalogu */dev*, przy pomocy BIOSu sprawdza istniejące w komputerze dyski. Plik konfiguracyjny GRUBa odczytywany jest każdorazowo przy starcie systemu dzięki czemu użytkownik nie musi modyfikować bootsektora przy każdej zmianie ustawień.

1.3 Funkcje systemu

Podstawą funkcją realizowanego serwera jest umożliwienie użytkownikom sieci dostępu do Internetu. W trosce o ich wygodę serwer będzie skonfigurowany tak aby wszystkie usługi działały tuż po podpięciu się do sieci, bez konieczności konfiguracji po

stronie użytkownika, niezależnie od systemu z jakiego korzysta. Zaprojektowany system będzie umożliwiał:

- filtrowanie pakietów w celu odrzucenia tych, które potencjalnie mogą stwarzać zagrożenie
- wymianę plików między użytkownikami w sieci lokalnej
- zdalne podłączanie się do sieci lokalnej z Internetu
- proste i intuicyjne zarządzanie serwerem przez przeglądarkę internetową dla osób uprawnionych
- monitorowanie ruchu w sieci

1.3.1 Filtrowanie pakietów

Do kierowania ruchem sieci na serwerze wykorzystany zostanie Iptables^[2]. Zawarty w nim mechanizm filtrowania pakietów będzie pełnił rolę zapory sieciowej i NATu (*Network Address Translation*). Każdy pakiet odebrany przez interfejs sieciowy sprawdzany będzie według zdefiniowanych przez administratora reguł. Przefiltrowany pakiet będzie zaakceptowany, odrzucony lub przekierowany dalej. Domyślnie wszystkie pakiety przychodzące i przekierowane będą blokowane a przepuszczone zostaną tylko te które pasują do którejś z określonych reguł.

Do serwera zostanie przypisany jeden publiczny adres IP^[3], który dzięki zastosowaniu technologii NAT współdzielony będzie między komputery w sieci lokalnej. NAT będzie zmieniał w pakietach adres IP oraz port tak aby wyglądało że pakiety wysyłane zostały z publicznego adresu IP serwera. Wszystkie zmiany będą zapamiętywane w celu ponownego ich wprowadzenia w pakietach powracających.

Ponadto Iptables w połączeniu z serwerem DHCP^[4] będzie monitorowało ruch w sieci. Na podstawie danych z DHCP o przydzielonych adresach IP będą powstawały w Iptables reguły zliczające ilość pobieranych i wysłanych danych.

1.3.2 Wymiana plików w sieci

Wymiana plików w sieci dostępna będzie dzięki zastosowaniu w systemie serwera Samby^[5]. Jest to szybkie, bezpieczne i darmowe oprogramowanie służące do współdzielenia plików i drukarek. Wykorzystuje ono protokół transmisji SMB dzięki czemu znakomicie współpracuje z klientami wyposażonymi w system Windows. Oprócz współdzielenia

zasobów Samba umożliwia również: wspomaganie klientów w przeglądaniu otoczenia sieciowego, uwierzytelnianie klientów za pomocą loginów i haseł, współpracę z domeną oraz odwzorowywanie nazw hostów jako serwer WINS. Niegdyś oprogramowanie to uznawane było za darmową alternatywę dla serwerów Windowsowych, obecnie stosowane jest jako znakomite narzędzie nieraz przewyższające możliwości i szybkość rozwiązań stosowanych przez Microsoft.

1.3.3 Wirtualna sieć prywatna

Zdalne podłączanie się do sieci lokalnej i przeglądanie jej zasobów jest możliwe dzięki zastosowaniu technologii VPN (*Virtual Private Network*) – Wirtualna sieć prywatna. Najbezpieczniejsze byłoby zastosowanie protokołu IPsec, jednak biorąc pod uwagę skomplikowaną konfigurację oraz brak domyślnie zainstalowanego klienta na systemach Windows nie jest to najlepszy wybór. Idealnym rozwiązaniem okazał się protokół PPTP^[6] (*Point to Point Tunneling Protocol*), który jest stosunkowo bezpieczny i łatwo konfigurowalny od strony klienta na większości systemów.

PPTP to protokół pozwalający na przesyłanie skompresowanych i zaszyfrowanych pakietów PPP na bazie protokołu IP. Do kompresji danych używany jest protokół MPPC (*Microsoft Point to Point Compression*) a do szyfrowania MPPE (*Microsoft Point to Point Encryption*) korzystający z algorytmu RC4. W czasie nawiązywania połączenia tworzącego tunel VPN wykorzystany zostanie mechanizm autoryzacji MS-CHAP w wersji drugiej pozwalający na wzajemne uwierzytelnienie za pomocą haseł zaszyfrowanych algorytmem DES.

1.3.4 Zarządzanie i monitoring

Zarządzanie serwerem z linii poleceń daje wiele możliwości ale jest mało wygodne. Wymaga od administratora dobrej znajomości składni poleceń oraz miejsca przechowywania skryptów i plików konfiguracyjnych. Bardzo przydatne okazują się nakładki graficzne oraz web serwisy pozwalające na szybkie i intuicyjne zarządzanie systemem.

Aby udostępnić administratorowi stronę www do zarządzania serwerem zostanie na nim zainstalowany serwer HTTP oraz interpreter języka PHP^[7] z trybie FastCGI. Jako serwer HTTP został wybrany szybki, zgodny ze standardami i łatwy w obsłudze Lighttpd^[8]. Zaprojektowano go z myślą o systemach wymagających wysokiej wydajności przy małym

zapotrzebowaniu na pamięć i moc obliczeniową procesora. Dzięki zastosowaniu protokołu FastCGI w przypadku wystąpienia błędów w PHP serwer HTTP nie ulega zawieszeniu ponieważ procesy te są od siebie odizolowane. Jako język do generowania serwisu www został wybrany PHP ponieważ jest on jednym z najlepiej udokumentowanych i wspieranych języków skryptowych.

Web serwis będzie pozwalał użytkownikom sieci na kontrolowanie ilości danych jakie pobrali i odebrali z Internetu. Administrator dodatkowo po zalogowaniu będzie mógł:

- zobaczyć dostępne interfejsy sieciowe wraz z ich konfiguracją
- podejrzeć dwadzieścia ostatnio zarejestrowanych zdarzeń z pliku */etc/log/messages*
- sprawdzić ostatnie logowania użytkowników na serwer
- włączać, wyłączać oraz restartować usługi dostępne na serwerze
- kontrolować dostępną ilość pamięci, wolnego miejsca na dysku oraz połączenie z Internetem

1.4 Wykorzystywany sprzęt

Do realizacji serwera został wybrany mały komputer biurowy firmy Dell z procesorem Pentium 4, przedstawiony na rysunku 1.2. Przy wyborze maszyny kierowano się głównie małymi wymiarami, cichą pracą, małym zużyciem energii oraz niską ceną.



Rys. 1.2. komputer Dell OptiPlex GX270

W tabeli 1.1 zaprezentowano szczegółowe dane techniczne komputera

Typ procesora	Intel Pentium 4, 2800 MHz
Zbiór instrukcji procesora	x86, MMX, SSE, SSE2

Nazwa płyty głównej	Dell OptiPlex GX270
Mikroukłady płyty głównej	Intel Springdale-G i865G, Intel 82801EB ICH5
Kontroler IDE	Intel(R) 82801EB Ultra ATA Storage Controllers
Karta wideo	Intel(R) 82865G Graphics Controller (96 MB)
Karta dźwiękowa	Analog Devices AD1981B(L) - AC'97
Dysk twardy	WDC WD400BB-75FJA1 (37 GB, IDE)
Napęd optyczny	LG CD-ROM CRN-8245B (24x CD-ROM)
Karta sieciowa 1	Intel(R) PRO/1000 MT Network Connection
Pamięć operacyjna	2x256 MB

Tab. 1.1 Dane techniczne serwera

Komputer w standardowej konfiguracji posiadał tylko jeden interfejs sieciowy co nie wystarczało do zrealizowania założeń pracy więc dokupiono dodatkową kartę sieciową i podczepiono ją do portu PCI komputera. Karta przedstawiona jest na rysunku 1.3



Rys. 1.3. karta sieciowa D-link DGE-530T

Dokupiono również ośmio-portowy przełącznik przedstawiony na rysunku 1.4, który będzie znajdował się w sieci lokalnej i pozwoli na podłączenie do serwera siedmiu hostów



Rys. 1.4. switch D-link DSG-1005D

1.5 Porównanie i wybór dystrybucji

Na świecie dostępnych jest wiele dystrybucji Linuksa, różnią się one od siebie dołączonym oprogramowaniem, sposobem instalacji, startem systemu, środowiskiem graficznym oraz przeznaczeniem. Możemy wyróżnić dystrybucje komercyjne, które producenci dostarczają wraz z obszerną dokumentacją i odpłatną pomocą techniczną oraz dystrybucje otwarte, które są tworzone głównie przez pasjonatów nie nastawionych na zysk. Wielkość dystrybucji jest bardzo zróżnicowana i zależy ona głównie od ilości dołączonego oprogramowania. Poniżej znajduje się opis kilku wybranych dystrybucji serwerowych, które brałem pod uwagę jako dystrybucję bazową do swojego projektu:

1.5.1 Gentoo

Gentoo to dystrybucja przeznaczona dla zaawansowanych użytkowników chcących dogłębnie poznać system. Dystrybucja zoptymalizowana jest dla konkretnej architektury więc system startuje szybko a jego wydajność jest na wysokim poziomie w porównaniu z innymi dystrybucjami. Gentoo dostarczane jest między innymi jako LiveCd z możliwością instalacji na dysku w trybie graficznym. Świeżo zainstalowany system zajmuje ponad 1,5 GB. Instalacja systemu trwa długo gdyż niemal całe oprogramowanie kompiluje się z kodu źródłowego, nie jest to jednak czas zmarnowany ponieważ dobrze postawione Gentoo działa długo i stabilnie.

W odróżnieniu od innych dystrybucji Gentoo nie opiera się o skompilowane paczki, lecz posiada własny system zarządzania pakietami (Portage), który kompiluje pakiety z kodu źródłowego.

Gentoo 2008.0 zawiera między innymi:

- Kernel 2.6.24
- Portage 2.1.4.4
- Xfce 4.4.2
- Gcc 4.1.2
- Glibc 2.6.1

1.5.2 Debian GNU/Linux

Jest to uniwersalna i jedna z najstarszych dystrybucji Linuksa charakteryzująca się wysoką stabilnością. Odznacza się dużą liczbą pakietów oraz znakomitą narzędziem do ich instalacji - APT. Z poziomu instalatora systemu można wybrać pakiety które będą potrzebne do działania serwera, dodatkowo można wspomóc się istniejącymi już profilami: serwer www, serwer druku, serwer DNS, serwer plików, serwer pocztowy, itd.

Dystrybucja nastawiona jest na bezpieczeństwo, wszystkie istotne pakiety łtane są na bieżąco a ich bezproblemowa instalacja odbywa się automatycznie po wydaniu jednego polecenia. System po zainstalowaniu jest w całości spolszczony co czasami ułatwia pracę. Na podstawie Debiana powstało wiele znanych dystrybucji, chociażby Ubuntu, które niezmiennie od ponad dwóch lat znajduje się na pierwszym miejscu w rankingu popularności w serwisie <http://distrowatch.com/>

Debian 4.0r4 zawiera między innymi:

- Kernel 2.6.24
- Iptables 1.3.6
- Apache 2.2.3
- OpenSSH 4.3p2

1.5.3 Vyatta

Vyatta to sieciowa dystrybucja Linuksa oparta na Debianie przystosowana do pełnienia funkcji routera i firewalla. Pozwala ona zbudować z klasycznego komputera router odpowiadający parametrami, funkcjonalnością, niezawodnością i wydajnością drogim routerom Cisco. Dystrybucja nie posiada interfejsu graficznego, ale wyposażona jest w prosty a jednocześnie dający wiele możliwości interfejs dostępny przez przeglądarkę internetową.

W systemie wbudowana jest obsługa szerokiej gamy funkcji, niestety podczas instalacji nie mamy możliwości ich wyboru. Dystrybucja obsługuje większość protokołów sieciowych takich jak RIP, OSPF i BGP, QoS, PPPoE, SNMP, DDNS, DHCP czy translacja adresów NAT. Dodatkowo dystrybucję można skonfigurować jako bramkę VPN z obsługą PPTP, L2TP i IPSec.

Możliwości Vyatta można przedstawić na przykładzie portalu <http://dobreprogramy.pl/> udostępniającego na szeroką skalę programy użytkowe. W 2008 roku portalowi przestał wystarczać router Cisco zdolny przetwarzać ruch 100 megabitów. Nowy gigabitowy router znacznie przekraczał możliwości finansowe redakcji więc jej pracownicy postanowili poszukać tańszego rozwiązania. Vyatta sprawdziła się w tej roli doskonale a redakcja zamiast wydawać ponad 50.000zł na router zakupiła serwer za niewiele ponad 2.000zł¹.

Vyatta 4.1 zawiera między innymi:

- Kernel 2.6.23
- Iptables 1.4
- OpenSSH 5.1p1

1.5.4 Trustix Secure Linux

Trustix to bezpieczna dystrybucja Linuksa, zoptymalizowana pod kątem pracy na serwerach. Zawiera ona starannie dobrane oprogramowanie, umożliwiające uruchomienie serwera poczty, www czy też bazy danych. Po domyślnej instalacji zajmuje niewiele ponad 100 MB i zawiera tylko i wyłącznie aplikacje niezbędne do działania systemu.

Dzięki dokładnemu dopasowaniu zależności pomiędzy pakietami podczas instalacji potrzebnego oprogramowania nie instaluje się duża ilość innych aplikacji, z których prawdopodobnie nigdy nie skorzystamy.

Ponieważ Trustix jest dystrybucją typowo serwerową, nie zawiera w sobie środowiska graficznego czy też dość popularnego menażera plików Midnight Commander. Nie jest to systemem dla początkujących administratorów. Pomimo łatwej do przeprowadzenia instalacji sama administracja systemem wymaga pewnej znajomości Linuksa.

Trustix 3.0 zawiera między innymi:

- Kernel 2.6.11.12 (z licznymi dodatkami zwiększającymi bezpieczeństwo)
- Apache 2.0.54
- IPTables 1.3.1
- MySQL 4.1.12
- OpenSSH 4.1p1
- PHP 5.0.4

¹ dane z publikacji <http://www.techit.pl/MoimZdaniem/View.aspx?2867>

- PostgreSQL 8.0.3
- Samba 3.0.14a

Początkowo to właśnie Trustix miał być dystrybucją bazową do mojej pracy, niestety wsparcie dla niej zostało zakończone w związku z czym postanowiłem zmienić swój wybór.

1.5.5 Fedora

Jest to dystrybucja ogólnego użytku, nadająca się zarówno na specjalistyczny serwer jak i na desktop do domu. Celem twórców było zbudowanie w pełni użytecznego, nowoczesnego systemu operacyjnego w oparciu o wolne oprogramowanie. Fedora jest tworzona przez firmę Red Hat i często określa się ją jako dystrybucję testową dla rozwiązań wprowadzanych później do Red Hat Enterprise Linux.

Dzięki bardzo dopracowanemu instalatorowi użytkownik może wybrać pakiety, które będą mu niezbędne do późniejszej pracy z systemem. Fedora jest to system całkowicie spolszczony a jego kolejne wydania pojawiają się często i zawierają najnowsze dostępne oprogramowanie, nawet jeśli nie są to wersje stabilne

Fedora 9 zawiera między innymi:

- Kernel 2.6.25
- Gnome 2.22
- KDE 4
- Obsługę systemu plików ext4

1.5.6 Core GNU/Linux

Jest to minimalistyczna dystrybucja Linuksa zawierająca jedynie programy niezbędne do działania systemu. Wersja przeznaczona dla zaawansowanych użytkowników Linuksa, którzy chcą zbudować własną dystrybucję od podstaw. Zawiera niemalże najnowsze wersje pakietów ale nie posiada automatycznej instalacji przez co wszystko wykonuje się ręcznie. W dystrybucji dostępny jest tylko tryb tekstowy.

Dystrybucja została wybrana przeze mnie jako podstawa do zrealizowania serwera więc szczegółowy opis instalacji i konfiguracji znajduje się w dalszej części pracy. Dystrybucja Core GNU/Linux 2.0 wymaga od użytkownika dobrej znajomości środowiska Linux oraz sporego wkładu pracy i czasu. Po zainstalowaniu zajmuje niewiele ponad 100 MB gdyż

posiada minimalną ilość pakietów i sterowników dzięki czemu użytkownik może zainstalować tylko wybrane komponenty, które będą mu przydatne.

1.6 Wybór systemu plików

Najbardziej popularnym i uniwersalnym systemem plików w Linuksie jest obecnie *ext3*. Swoją popularność zawdzięcza prostocie i stosunkowo dużej wydajności. Wspiera on wszystkie elementy systemu plików Unix, takie jak: dowiązania symboliczne, pliki specjalne, prawa dostępu i wiele innych. Jest stabilny i dobrze przetestowany, a na rynku istnieje szereg programów skutecznie naprawiających ten system plików po awarii. W porównaniu z *ext2* dodano w nim mechanizm księgowania zmniejszający niebezpieczeństwo utraty danych oraz skracający do minimum czas sprawdzania systemu plików po awarii. System plików *ext3* jest kompatybilny wstecz z *ext2* dzięki czemu istnieje możliwość montowania systemu *ext3* jako *ext2* oraz łatwej i bardzo bezpiecznej migracji z *ext2* na *ext3* nawet bez odmontowywania systemu plików.

Systemy Linux bardzo wydajnie wykorzystują pamięć operacyjną i SWAP jest rzadko używany. Sprawdza się to w przypadku systemów biurkowych, jednak serwery rządzą się swoimi prawami. W każdej chwili mogą one być poddawane nieoczekiwanym przeciążeniom, takim jak niekontrolowany proces, atak DoS czy nawet efekt Slashdot. Pamięć SWAP ma jednak swoje ograniczenia, na komputerze z procesorem x86 limit wynosi 2 GB.

1.7 Założenia projektu

Celem przedstawianej pracy jest zaprojektowanie oraz zrealizowanie serwera sieci lokalnej, którego podstawą jest jedna z dystrybucji Linuksa. Wybrano dystrybucję, która posiadała minimalną ilość zainstalowanych programów, dzięki czemu można zainstalować tylko te, które są niezbędne do prawidłowej pracy serwera. Zmniejsza to jego obciążenie, czas aktualizacji oraz zwiększa bezpieczeństwo i porządek w systemie dzięki możliwie małej ilości uruchomionych usług.

Projektowany system ma za zadanie w znacznym stopniu usprawnić pracę administratora, zwiększyć wydajność oraz bezpieczeństwo sieci, ułatwić jej monitorowanie a także dostarczyć usługi wymienione w podrozdziale 1.3.

2. Realizacja serwera

Aby opisany w podrozdziale 1.4 komputer spełniał wcześniej wymienione założenia zainstalowano w nim system Core 2.0. Specyfika użytej dystrybucji wymagała autorskiej selekcji, konfiguracji i kompilacji wybranych składników systemu. W ramach projektu wykonano następujące czynności:

- przeprowadzono selekcję oprogramowania do instalacji, poprzedzoną analizą potrzeb systemu i analizą możliwości dostępnego oprogramowania
- skonfigurowano bootloader GRUB aby móc uruchamiać zainstalowany system
- skompilowano jądro w celu dodania obsługi urządzeń oraz nowych funkcji
- zainstalowano program Mcron^[9], który cyklicznie będzie wykonywał wskazane mu zadania
- zainstalowano program klienta protokołu DHCP odpowiedzialny za konfigurację interfejsu *eth0*
- napisano skrypt kontrolujący stan połączenia serwera z Internetem
- skonfigurowano interfejs sieciowy *eth1* oraz zainstalowano i skonfigurowano serwer DHCP przydzielający adresy IP hostom w sieci lokalnej
- zainstalowano i skonfigurowano serwer SSH^[10] umożliwiający na zdalne podłączenie się i zarządzanie serwerem
- zainstalowano program Iptables i stworzono szereg reguł filtrujących pakiety
- napisano skrypty do zliczania ruchu renerowanego przez hosty w sieci lokalnej
- zainstalowano serwer HTTP oraz interpretator języka PHP aby możliwe było uruchamianie skryptów napisanych w języku PHP
- zainstalowano i skonfigurowano program Samba, który umożliwia wymianę plików między użytkownikami sieci
- zainstalowano i skonfigurowano program Poptop dostarczający możliwość zdalnego podłączania się do sieci lokalnej za pomocą technologii VPN
- stworzono skrypty umożliwiające łatwe zarządzanie usługami
- używając języka PHP napisano skrypty przedstawiające w czytelny sposób ruch w sieci oraz umożliwiające zarządzanie serwerem przez przeglądarkę

2.1 Instalacja Core 2.0

Zajmującą około 150 MB dystrybucję Core 2.0 GNU/Linux pobrano ze strony autorów <http://coredistro.org/>. Po wystartowaniu komputera z płyty uruchamia się system, co umożliwia jego instalację na dysku.

Mając do dyspozycji dysk `/dev/hda` o pojemności 40 GB stworzono dwie partycje podstawowe:

- `/dev/hda1` o wielkości 38 GB i przeznaczono ją na główny system plików
- `/dev/hda2` o wielkości 2 GB i przeznaczono ją na pamięć SWAP

Podczas tworzenia partycji korzystano z programu do zarządzania dyskami *fdisk*

```
$ fdisk /dev/hda
```

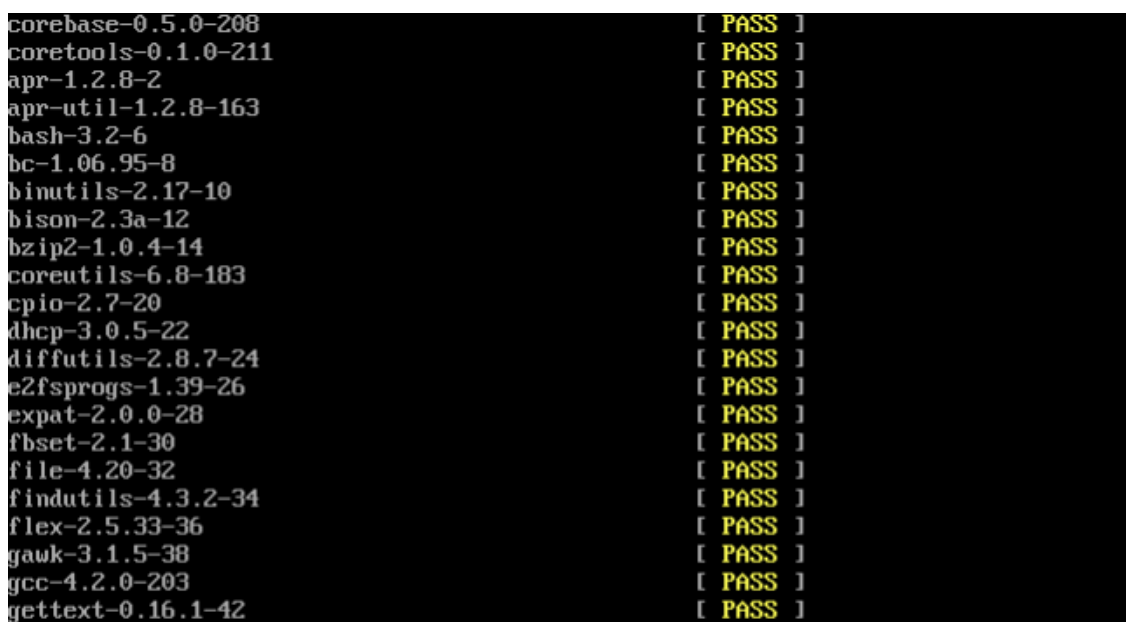
Na partycji `/dev/hda1` utworzono system plików *ext3*

```
$ mkfs -t ext3 /dev/hda1
```

Po utworzeniu system plików podmontowano partycję w katalogu `/mnt` poczym instalowano tam wcześniej wybraną dystrybucję systemu Linuks. Proces instalacji przedstawiono na rysunku 2.1

```
$ mount /dev/hda1 /mnt -t ext3
```

```
$ install_core /mnt
```



```
corebase-0.5.0-208      [ PASS ]
coretools-0.1.0-211    [ PASS ]
apr-1.2.8-2            [ PASS ]
apr-util-1.2.8-163     [ PASS ]
bash-3.2-6            [ PASS ]
bc-1.06.95-8          [ PASS ]
binutils-2.17-10      [ PASS ]
bison-2.3a-12         [ PASS ]
bzip2-1.0.4-14        [ PASS ]
coreutils-6.8-183     [ PASS ]
cpio-2.7-20           [ PASS ]
dhcp-3.0.5-22         [ PASS ]
diffutils-2.8.7-24    [ PASS ]
e2fsprogs-1.39-26     [ PASS ]
expat-2.0.0-28        [ PASS ]
fbset-2.1-30          [ PASS ]
file-4.20-32          [ PASS ]
findutils-4.3.2-34    [ PASS ]
flex-2.5.33-36        [ PASS ]
gawk-3.1.5-38         [ PASS ]
gcc-4.2.0-203         [ PASS ]
gettext-0.16.1-42     [ PASS ]
```

Rys 2.1. zrzut ekranu ilustrujący przebieg procesu instalacji systemu Core 2.0 GNU/Linux

Zainstalowany system operacyjny znajduje się w katalogu */mnt*. W celu skonfigurowania niezbędnych ustawień zmieniono główny system plików na */mnt* wcześniej podmontowując potrzebne do działania katalogi urządzeń i procesów. Zmiana systemu plików jest możliwa, ponieważ system operacyjny zainstalowany w */mnt* jest taki sam jak ten z którego wystartowano

```
$ mount -t proc none /mnt/proc
```

```
$ mount -o bind /dev /mnt/dev
```

```
$ chroot /mnt /bin/bash --login
```

Po przełączeniu na nowy system plików dodano wcześniej utworzone partycje do pliku konfiguracyjnego */etc/fstab*, zawierającego informacje na temat znajdujących się w systemie partycji, oraz sposobu, w jaki mają być montowane

```
$ echo "/dev/hda1 / ext3 defaults 0 0" >> /etc/fstab
```

```
$ echo "/dev/hda2 none swap defaults 0 0" >> /etc/fstab
```

W celu zabezpieczenia serwera przed niepożądanym dostępem ustawiono silne hasło dla konta *root* za pomocą polecenia *passwd*

```
$ passwd
```

Ustawiono również strefę czasową w systemie, tworząc dowiązanie symboliczne z nazwą miasta, w której znajduje się serwer

```
$ ln -sf /usr/share/zoneinfo/Europe/Warsaw /etc/localtime
```

Ostatnią rzeczą jaką zrobiono aby nowo zainstalowany system wystartował po włączeniu komputera jest konfiguracja bootloader'a.

2.2 Konfiguracja GRUB'a

Dystrybucja Core GNU/Linux w wersji 2.0 dostarczona jest z GRUBem, niestety nie jest on skonfigurowany. W pierwszej kolejności stworzyłem plik *device.map* opisujący relację pomiędzy Linuksowym a GRUBowym nazewnictwem napędów. Następnie zainstalowałem GRUBa w bootsektorze dysku, na którym znajduje się dystrybucja Linuksa, podając jednocześnie ścieżkę do pliku konfiguracyjnego. Kończąc proces instalacji wskazałem GRUBowi jego katalog z plikami oraz ścieżkę do pliku z odwzorowaniem urządzeń

```
$ grub-mkdevicemap
```

```
$ grub-install --grub-setup=/boot/grub/grub.cfg /dev/hda
```

```
$ grub-setup --directory=/boot/grub --device-map=/boot/grub/device.map /dev/hda
```

W pliku konfiguracyjnym */boot/grub/grub.cfg* ustawiono następujące parametry

set timeout=2 – czas w sekundach jaki bootloader czeka na wybranie systemu przez użytkownika ustawiono na możliwie krótki, żeby system podczas restartu jak najszybciej wrócił do pełnienia swoich funkcji

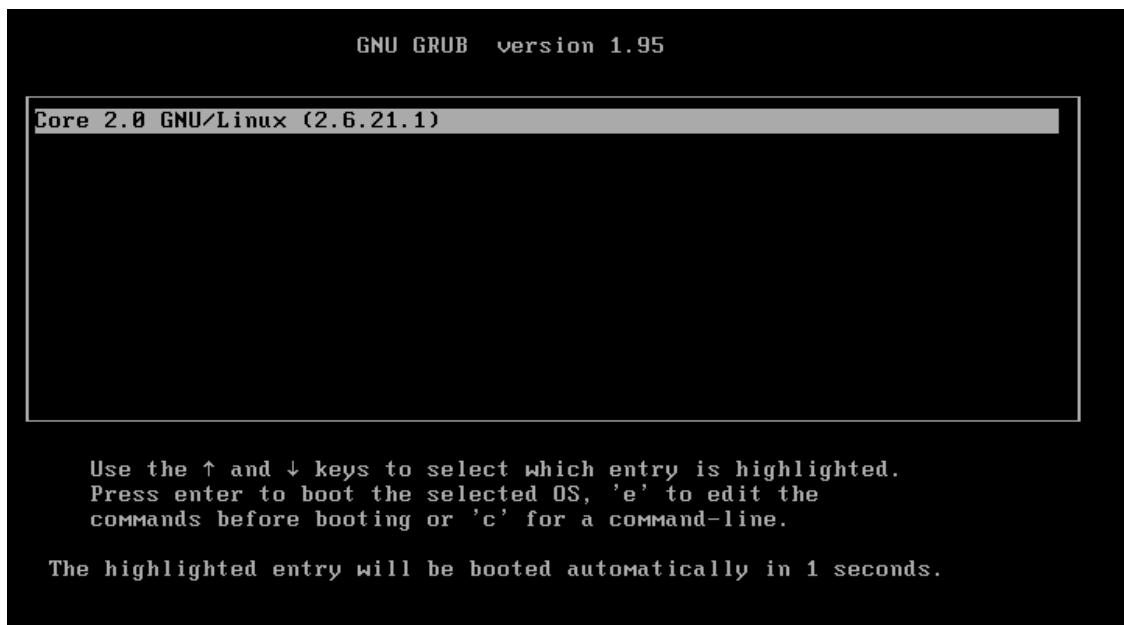
set default=0 – ustawiono numer domyślnego obrazu jaki zostanie załadowany jeśli użytkownik nie dokona wyboru

Stworzono sekcję z nazwą systemu oraz wersją jądra

```
menuentry "coredistro (2.6.21.1)" {
```

```
    set root=(hd0, 1) – wskazano partycję na której zawarte są pliki GRUBa
```

```
    linux=(hd0, 1)/boot/vmlinuz root=/dev/hda1 vga=5 – określono ścieżkę do pliku z obrazem jądra, wskazano położenie głównego systemu plików oraz tryb graficzny  
}
```



Rys. 2.2. zrzut ekranu ilustrujący okno wyboru systemu w programie GRUB

2.3 Kompilacja jądra

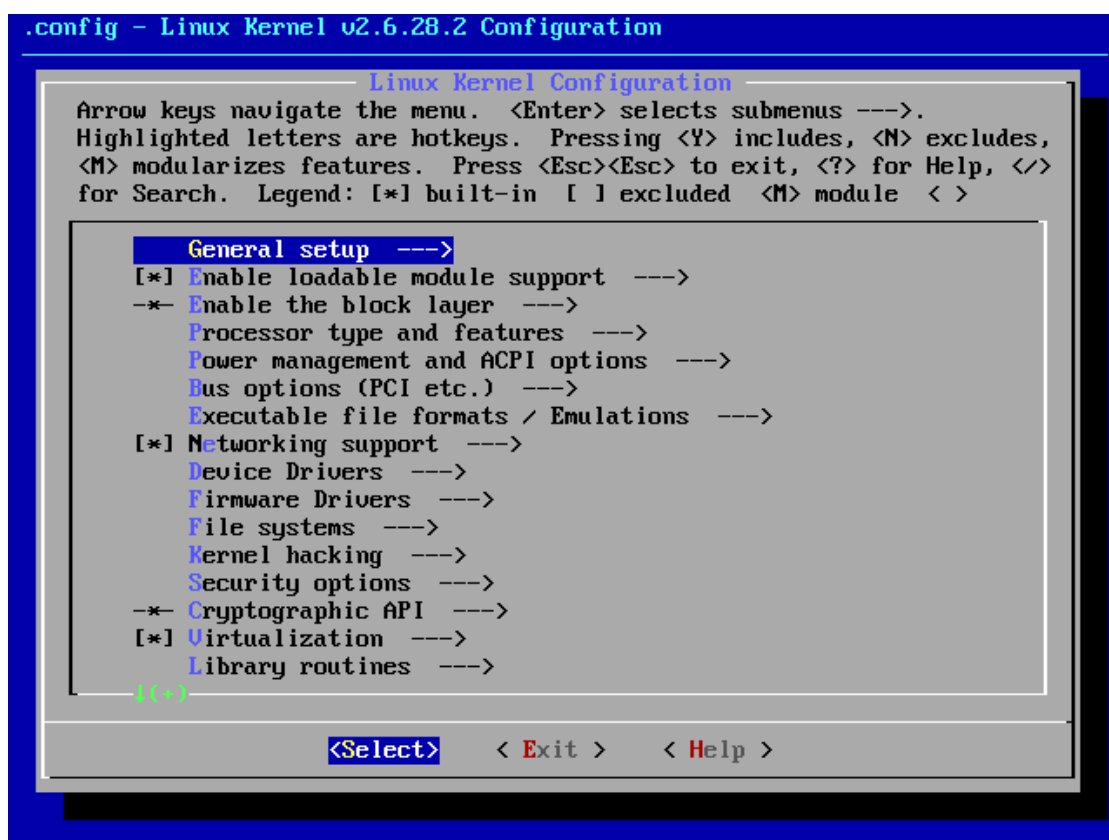
Najnowsze źródła jądra pobierano z oficjalnej strony projektu <http://kernel.org/>, aktualnie jest to wersja 2.6.28.2. Archiwum ze źródłami umieszczono w katalogu `/usr/src` i rozpakowowano je

```
$ tar jxvf linux-2.6.28.2.tar.bz2
```

```
$ cd linux-2.6.28.2
```

System nie jest wyposażony w tryb graficzny więc podczas konfiguracji skorzystano z polecenia `menuconfig` do wybrania potrzebnych funkcji co pokazane jest na rysunku 3.3

```
$ make menuconfig
```



Rys 2.3 zrzut ekranu ilustrujący konfigurację jądra za pomocą `menuconfig`

W najnowszym jądrze znajduje się ponad kilkaset funkcji, więc opiszę tylko najważniejsze z nich oraz te, które przeznaczyłem do wkompiłowania w jądro bądź zainstalowania jako moduł:

[*] Support for paging of anonymous memory – włącza obsługę partycji wymiany (SWAP) dając tym samym możliwość tymczasowego przechowywania danych gdy zabraknie pamięci RAM

[*] System V IPC – włącza wewnętrzny proces komunikacyjny umożliwiający wymianę informacji między procesami. Jeśli opcja nie zostanie włączona część programów może działać niepoprawnie

[*] Enable loadable module suport – włącza obsługę modułów. Nawet jeśli żadna opcja nie zostanie zainstalowana jako moduł warto aktywować tę funkcję, żeby w przyszłości można było rozbudować jądro o nową opcję lub sterownik

[*] Module unloading - umożliwia usuwanie modułów z pamięci, jeśli nie włączy się tej opcji, wszystkie moduły załadowane do pamięci będą w niej przechowywane do zamknięcia systemu

[*] Forced module unloading – umożliwia natychmiastowe usunięcie modułu z pamięci, bez czekania aż przestanie on być używany nawet jeśli jądro uzna tę operację za niebezpieczną

<*> Anticipatory I/O scheduler – ustawia dany algorytm jako domyślny. Algorytm ten odpowiedzialny jest za szeregowanie operacji wejścia i wyjścia. Jego zadaniem jest zminimalizowanie liczby przeskoków mechanizmu służącego do odczytywania i zapisywania danych w urządzeniach blokowych. Jest to algorytm przedkładający operację odczytu nad operacją zapisu

[*] Machine Check Exception - włącza obsługę wyjątków w systemie umożliwiających tym samym odbieranie informacji od procesora dotyczących zaistniałych błędów. W zależności od wagi błędu system może wyświetlić błąd lub wyłączyć maszynę. Opcja działa na procesor Pentium lub nowszy

[*] PCI support – włącza obsługę urządzeń podłączonych przez interfejs PCI

[*] Unix domain sockets – włącza obsługę gniazd Unixowych. Opcja ta jest wykorzystywana przez procesy do komunikowania się między sobą. Wiele aplikacji nie zadziała bez włączenia tej opcji, na przykład syslog

[*] TCP/IP networking – włącza obsługę protokołu TCP/IP oraz tworzy interfejs wirtualny loopback

[*] IP: advanced router – włącza zaawansowane opcje routingu umożliwiające przekazywanie i redystrybucję pakietów między interfejsami

[*] IP: DHCP support – umożliwia obsługę protokołu DHCP

[*] IP: GRE tunnels over IP – sterowniki do obsługi tunelowania używające protokołu GRE niezbędne do działania usługi pptp (vpn)

[*] IPv4 Connection tracking – opcja umożliwia zapamiętywanie, jakie pakiety są przesyłane przez interfejsy. Opcja jest niezbędna do korzystania z usługi NAT, która została opisana w pierwszym rozdziale pracy

[*] Netfilter Xtables support – włącza obsługę filtra pakietów iptables

[*] Full NAT – umożliwia korzystanie z opcji masquerade oraz port forwarding

[*] MASQUARADE target support – włącza możliwość zmiany źródłowych lub docelowych adresów IP oraz portów TCP/UDP podczas przepływu pakietów

[*] ATA disk support – obsługa dysków podłączonych przez interfejs ATA

[*] Include IDE/ATAPI CDROM support – obsługa napędów optycznych podłączonych przez interfejs ATA

[*] Intel PIIX/ICH chipset suport – sterownik do chipu zainstalowanego na płycie głównej, umożliwiającą optymalną skonfigurację

[*] Inter® PRO/1000 Gigsbit Ethernet support – sterowniki do karty sieciowej zintegrowanej na płycie głównej

[*] New SysKonnnect GigaEthernet support - sterowniki do dodatkowej karty sieciowej D-link DGE-530T

<*> PPP (point-to-point protocol) support – włącza obsługę protokołu PPP umożliwiającego zestawianie tuneli punkt-punkt między hostami

[*] Hangcheck timer – włącza opcję wykrywającą zawieszenia zegara w systemie. Po wykryciu takiego błędu program może zresetować komputer bądź wyświetlić ostrzeżenie

[M] Support for Host-side USB – włącza obsługę urządzeń USB. Jeśli do komputera nie ma urządzeń podłączonych na stałe za pomocą interfejsu USB warto zainstalować tę opcję jako moduł

[*] UHCI HCD (most Intel and VIA) support – włącza obsługę kontrolera *Universal Host Controller Interface* – odpowiadającego za obsługę interfejsu USB. Kontroler ten jest najczęściej stosowany w urządzeniach produkowanych przez firmy Intelu oraz VIA

[M] USB Mass Storage support – włącza obsługę pamięci masowej umożliwiając korzystanie z pendrive'ów oraz zewnętrznych dysków na USB. Opcja bardzo przydatna w

początkowej fazie instalacji systemu do przenoszenia źródłem jądra oraz innych programów kiedy interfejsy sieciowe nie są jeszcze skonfigurowane

[M] SCSI disk support – włącza obsługę urządzeń z interfejsem SCSI. Nawet jeśli komputer nie jest wyposażony w taki interfejs to warto włączyć tę opcję gdyż większość urządzeń pamięci przenośnych jest rozpoznawanych przez system właśnie jako urządzenia SCSI

[*] Ext3 journalling file system support – włącza obsługę systemu plików *ext3*

[M] VFAT (Windows 95) fs suport – obsługa windowsowego systemu plików FAT, który jest najczęściej stosowanym systemem plików na urządzeniach przenośnych

[*] /proc file system support – włącza obsługę wirtualnego systemu plików */proc*, który dostarcza informacje o stanie systemu oraz pozwala na modyfikacje niektórych ustawień. Przed rozpoczęciem pracy z tym systemem należy go najpierw zamontować

[*] Enable verbose x86 bootup info messages – wyświetla informacje o dekompresji w czasie bootowania systemu. Jeśli opcja zostanie wyłączona to widoczne będą tylko błędy

Po skonfigurowaniu jądra i zapisaniu ustawień wszystkie dokonane wybory zostaną domyślnie zapamiętane w pliku *.config*. Następnie przystąpiono do właściwej kompilacji

```
$ make bzImage
```

Jądro skompilowane na podstawie wyżej opisanej konfiguracji ma rozmiar zaledwie 1,3 MB. Znajduje się ono w pliku */usr/src/linux-2.6.28.2/arch/i386/boot/bzImage*. Pomimo swoich niewielkich rozmiarów zawiera wszystkie niezbędne funkcje do pełnienia roli serwera sieciowego. Wybrana została modułarna postać jądra więc skompilowano i zainstalowano również moduły

```
$ make modules
```

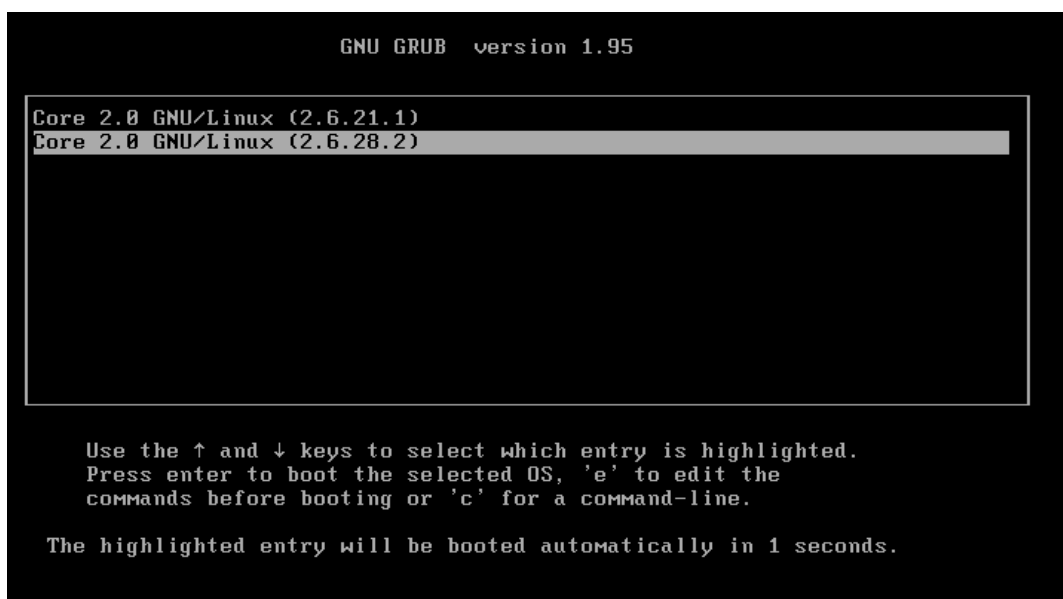
```
$ make modules_install
```

Wszystkie zainstalowane moduły znajdują się w katalogu */lib/modules/2.6.28.2/*. Przed wystartowaniem systemu z nowego jądra skopiowano jego obraz do katalogu */boot* oraz odpowiednio skonfigurowano bootloader. Plik z obrazem jądra jest nazywany umownie *vmlinuz*. Ponieważ w katalogu startowym jest wcześniejsza wersja jąder to nowe jądro oznaczono dodatkowo numerami wersji. Nie skasowano ani nie nadpisano starego jądra, ponieważ system z nowo utworzonym jądrem może się nie uruchomić

```
$ cp /usr/src/linux-2.6.28.2/arch/i386/boot/bzImage /boot/vmlinuz-2.6.28.2
```

Do pliku konfiguracyjnego bootloadera dodano sekcję dotyczącą nowego jądra, nie usuwając sekcji odpowiedzialnej za wystartowanie systemu z wcześniejszej wersji jądra na wypadek gdyby okazało się, że nowe jądro nie posiada wszystkich niezbędnych funkcji do wystartowania systemu

```
menuentry "coredistro (2.6.28.2)" {  
    set root=(hd0, 1)  
    linux=(hd0, 1)/boot/vmlinuz-2.6.28.2 root=/dev/hda1 vga=5  
}
```



Rys 2.4. zrzut ekranu ilustrujący okno wyboru systemu w programie GRUB

2.4 Mcron

Do cyklicznego wykonywania zadań w systemie został wykorzystany Mcron. Jest to następca programu Cron, rozpowszechniany na licencji GNU^[11]. Najnowsze archiwum ze źródłami Mcrona pobrano z oficjalnej strony projektu GNU <http://gnu.org/software/mcron/>, umieszczono w katalogu `/usr/src/`, rozpakowano, skompilowano i zainstalowano

```
$ tar -xvzf mcron-1.0.4.tar.gz
```

```
$ cd mcron-1.0.4
```

```
$ ./configure
```

```
$ make
```

```
$ make install
```

Mcron instaluje się domyślnie w katalogu */usr/local/bin/*. Przed pierwszym uruchomieniem aplikacji stworzono plik */etc/crontab* oraz katalog */var/cron/tabs/*, które są niezbędne do pracy programu

```
$ touch /etc/crontab
$ mkdir -p /var/cron/tabs/
$ /usr/local/bin/cron -n
```

2.5 Konfiguracja sieci

W serwerze zainstalowane są dwie gigabitowe karty sieciowe, jedna (*Intel PRO/1000 Gigabit Ethernet*) zintegrowana na płycie głównej, zaś druga (*D-link DGE-530T*) zamontowana jest w slotcie PCI. Wydając poniższe poleceni sprawdzono czy wszystkie interfejsy sieciowe są prawidłowo rozpoznane oraz jak zostały nazwane

```
$ dmesg | grep eth
```

W przypadku omawianej maszyny system prawidłowo rozpoznał dwa interfejsy

```
e1000: eth0: e1000_probe: Intel(R) PRO/1000 Network Connection
skge eth1: addr 00:19:5b:7c:c4:f5
```

Interfejs eth0 odpowiedzialny jest za komunikację serwera z Internetem, podłączony jest do niego modem dostarczony przez providera usług internetowych. Publiczny adres IP oraz pozostałe parametry protokołu TCP/IPv4 dostarczane są przez protokół DHCP.

2.5.1 Klient DHCP

Wybrana dystrybucja nie zawiera klienta DHCP, dlatego też trzeba go zainstalować. Najbardziej popularną aplikacją tego typu jest *dhcpcd*, którą można pobrać ze strony <http://phystech.com/>.

Konfiguracja aplikacji wymaga podania w opcji *build* architektury systemu

```
$ ./configure --build=i686-pc-linux-gnu
$ make
$ make install
```

Po zainstalowaniu aplikacja znajduje się w */usr/local/sbin/dhcpcd*, uruchamia się ją przez podanie w argumencie nazwy interfejsu

```
$ /usr/local/sbin/dhcpcd eth0
```

Jeśli aplikacja prawidłowo pobierze dane protokołu TCP/IPv4^[12] to konfiguruje interfejs eth0 i przechodzi do pracy w tle zapisując swój *pid* w pliku */etc/dhcp/dhcpd-eth0.pid*

W celu sprawdzenia czy interfejs został włączony oraz prawidłowo skonfigurowany użyto polecenia *ifconfig*, które wypisuje na ekran wszystkie działające w systemie interfejsy sieciowe wraz z ich parametrami

\$ ifconfig

```
root@localhost:~# $ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:E4:3D:C0
          inet addr:89.74.169.240  Bcast:255.255.255.255  Mask:255.255.255.0
          UP BROADCAST NOTRAILERS RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7048 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2148 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:595709 (581.7 Kb)  TX bytes:214273 (209.2 Kb)
          Interrupt:5 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:79 (79.0 b)  TX bytes:79 (79.0 b)
```

Rys. 2.5. zrzut ekranu ilustrujący konfigurację interfejsów sieciowych na serwerze

Do interfejsu został przypisany adres IP a szczegółowe dane pobrane z serwera DHCP zostały zapisane w pliku */etc/dhcp/dhcpd-eth0.info*

\$ cat /etc/dhcp/dhcpd-eth0.info

```
root@localhost:~# $ cat /etc/dhcp/dhcpd-eth0.info
IPADDR=89.74.169.240
NETMASK=255.255.255.0
NETWORK=89.74.169.0
BROADCAST=255.255.255.255
GATEWAY=89.74.169.1
DOMAIN='chello.pl'
DNS=62.179.1.62,62.179.1.63
DHCPSSID=10.128.1.64
DHCPGIADDR=89.74.89.1
DHCPPIADDR=10.128.1.64
DHCPCHADDR=00:0C:29:E4:3D:C0
DHCPCHADDR=00:14:F1:EB:A1:05
DHCPNAME=''
LEASETIME=3600
RENEWALTIME=1800
REBINDTIME=3150
INTERFACE='eth0'
CLASSID='Linux 2.6.28.2 i686'
CLIENTID=00:0C:29:E4:3D:C0
```

Rys. 2.6. zrzut ekranu ilustrujący zawartość pliku */etc/dhcp/dhcpd-eth0.info*

Dhcpd oprócz nadawania parametrów protokołu TCP/IP interfejsom dodaje do tablicy routingu adres bramy domyślnej oraz podmienia plik */etc/resolv.conf*, w którym zapisane są adresy serwerów DNS

\$ route

```

root@localhost:~$ route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
89.74.169.0 * 255.255.255.0 U 0 0 0 0 eth0
127.0.0.0 * 255.0.0.0 U 0 0 0 0 lo
default chello889874169 0.0.0.0 UG 0 0 0 0 eth0

```

Rys. 2.7 zrzut ekranu ilustrujący tablicę routingu

`$ cat /etc/resolv.conf`

```

root@server:/ $ cat /etc/resolv.conf
nameserver 194.204.152.34
nameserver 194.204.159.1

```

Rys. 2.8 zrzut ekranu ilustrujący adresy IP serwerów DNS

W celu zapewnienia jak najwyższej dostępności Internetu w sieci lokalnej, napisano skrypt `/usr/sbin/gwping`, który kontroluje połączenie serwera z Internetem. W przypadku braku połączenia kilka razy pod rząd serwer zrestartuje usługę `dhcpcd`, która odpowiedzialna jest za konfigurację interfejsu `eth0`. W przypadku, gdy restart usługi nie przywróci połączenia to resetowany jest system

`$ nano /usr/sbin/gwping`

```

GW=`route | grep default | awk '{print $2}'`
COUNT=`ping -c2 $GW | grep 'received' | awk -F',' '{ print $2}' | awk '{
print $1}'`

if [ $COUNT -gt 0 ]; then
    echo 0 > /var/cache/badgwping
else
    BADPING=`cat /var/cache/badgwping`
    echo ${BADPING+1} > /var/cache/badgwping

    if [ $BADPING -gt 5 ]; then
        /etc/init.d/dhcpcd restart
    fi

    if [ $BADPING -gt 15 ]; then
        echo 1 > /var/cache/badgwping
        reboot
    fi

```

Listing 2.1 zawartość pliku `/usr/sbin/gwping`

`$ chmod +x /usr/sbin/gwping`

Aby skrypt sprawdzał połączenie co minutę dodano go do `/etc/crontab`

```

GNU nano 2.0.5 File: /etc/crontab Modified
* * * * * root /usr/sbin/gwping

```

Rys. 2.9 zrzut ekranu ilustrujący zawartość pliku `/etc/crontab`

2.5.2 Ręczna konfiguracja protokołu TCP/IP

Eth1 to interfejs podłączony do switcha sieci lokalnej, wszystkie parametry protokołu TCP/IPv4 ustawiane są ręcznie. Jako adres sieci lokalnej przydzielono *192.168.1.0* z maską podsieci *255.255.255.0*. Realizowany serwer będzie pełnił rolę routera oraz bramy domyślnej dla sieci lokalnej więc zgodnie z przyjętymi zasadami nadano mu ostatni adres z danej podsieci, w tym przypadku jest to *192.168.1.254*

```
$ ifconfig eth1 inet 192.168.1.254 netmask 255.255.255.0
```

Za pomocą polecenia *ifconfig* sprawdzono czy interfejs skonfigurował się poprawnie

```
$ ifconfig
```

```
root@localhost:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:E4:3D:C0
          inet addr:89.74.169.240  Bcast:255.255.255.255  Mask:255.255.255.0
          UP BROADCAST NOTRAILERS RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:497 errors:0 dropped:0 overruns:0 frame:0
          TX packets:19 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:44302 (43.2 Kb)  TX bytes:5394 (5.2 Kb)
          Interrupt:5 Base address:0x2000

eth1      Link encap:Ethernet  HWaddr 00:0C:29:E4:3D:CA
          inet addr:192.168.1.254  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:35 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:1470 (1.4 Kb)
          Interrupt:5 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:224 (224.0 b)  TX bytes:224 (224.0 b)
```

Rys. 2.10. zrzut ekranu ilustrujący konfigurację interfejsów sieciowych na serwerze

Siec *192.168.1.0* została automatycznie dodana do tablicy routingu ponieważ jest bezpośrednio podłączona do jednego z interfejsów

```
$ route
```

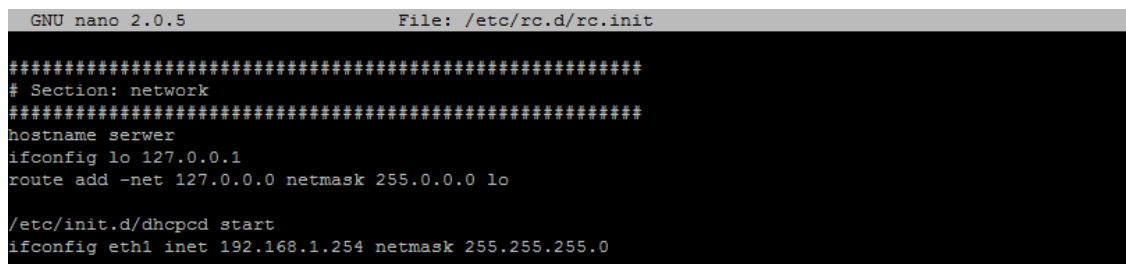
```
root@localhost:~# route
Kernel IP routing table
Destination        Gateway            Genmask           Flags Metric Ref    Use Iface
192.168.1.0        *                  255.255.255.0     U        0      0        0 eth1
89.74.169.0        *                  255.255.255.0     U        0      0        0 eth0
127.0.0.0          *                  255.0.0.0         U        0      0        0 lo
default            chello089074169  0.0.0.0           UG        0      0        0 eth0
```

Rys. 2.11. zrzut ekranu ilustrujący tablicę routingu

Tak skonfigurowany serwer bez problemu łączy się z urządzeniami w sieci lokalnej oraz w Internecie. Niestety po restarcie systemu cała konfiguracja sieci zostanie utracona,

dlatego też dodano odpowiednie wpisy w sekcji *network* pliku */etc/rc.d/rc.init*. W opisywanym pliku ustawiono również nazwę maszyny na *serwer*

```
$ nano /etc/rc.d/rc.init
```



```
GNU nano 2.0.5 File: /etc/rc.d/rc.init
#####
# Section: network
#####
hostname server
ifconfig lo 127.0.0.1
route add -net 127.0.0.0 netmask 255.0.0.0 lo

/etc/init.d/dhcpd start
ifconfig eth1 inet 192.168.1.254 netmask 255.255.255.0
```

Rys. 2.12. zrzut ekranu ilustrujący sekcję sieci w pliku */etc/rc.d/rc/init*

2.5.3 Serwer DHCP

Najnowsze źródła demona DHCP można pobrać ze strony <http://isc.org/>. Serwer połączony jest z Internetem więc posłużono się poleceniem *wget*, które pobiera pliki bezpośrednio z Internetu na realizowany serwer

```
$ wget http://ftp.isc.org/isc/dhcp/dhcp-4.1.0.tar.gz
```

```
$ tar -zxvf dhcp-4.1.0.tar.gz
```

```
$ cd dhcp-4.1.0
```

Podczas konfiguracji wyłączono obsługę protokołu IPv6 ponieważ w systemie nie jest on zaimplementowany i uruchomienie serwera DHCP byłoby niemożliwe

```
$ ./configure --disable-dhcpv6
```

```
$ make
```

```
$ make install
```

Program instaluje się domyślnie w */usr/sbin/dhcpd*. W pliku konfiguracyjnym */etc/dhcpd.conf* ustawiono adres IP sieci lokalnej w której będzie działał serwer na 192.168.1.0 oraz maskę podsieci na 255.255.255.0. Dodatkowo wskazano klientom DHCP bramę domyślną oraz adres DNS serwera.

Poniżej głównej konfiguracji DHCP dopisano wpis odnoszący się do serwera windowsowego zawierający jego adres MAC oraz IP dzięki czemu będzie on miał statycznie przydzielone IP z sieci lokalnej co w dalszej części pracy przyda się do przekierowania odpowiedniego ruchu na ten serwer.

```
#sieć / maska
subnet 192.168.1.0 netmask 255.255.255.0 {
    #zakres adresow
    range 192.168.1.10 192.168.1.253;

    #domena
    option domain-name "przeor.eu";

    #czas dzierżawy
    default-lease-time 5529600;

    #serwer dns
    option domain-name-servers 194.204.159.1;

    #broadcast
    option broadcast-address 192.168.1.255;

    #brama
    option routers 192.168.1.254;
}

#klienci statyczni
host serwer_windows {
    hardware ethernet 00-1D-60-D8-13-08;
    fixed-address 192.168.1.1;
}
```

Listing 2.2 zawartość pliku /etc/dhcpd.conf

Uruchamianie usługi zostanie opisane w rozdziale 2.7 ponieważ zostało połączone z działaniem *Iptables*

2.6 OpenSSH

Do zainstalowania SSH niezbędny jest zestaw bibliotek implementujący protokół SSL oraz mechanizmy kryptograficzne – OpenSSL. Najnowszą ich wersję pobrano ze strony <http://openssl.org/> i zainstalowano

```
$ wget http://openssl.org/source/openssl-0.9.8j.tar.gz
$ tar -zxvf openssl-0.9.8j.tar.gz
$ cd openssl-0.9.8j
$ ./configure
$ make
$ make install
```

Po pomyślnym zainstalowaniu bibliotek OpenSSL przystąpiono do pobrania OpenSSH z oficjalnej strony projektu <http://openssh.com/>

```
$ wget ftp://ftp.openbsd.org/pub/OpenBSD/OpenSSH/openssh-5.2.tar.gz
$ tar -zxvf openssh-5.1.tar.gz
```

```
$ cd openssh-5.1p
```

Domyślnie aplikacja korzysta z plików konfiguracyjnych umieszczonych w katalogu */usr/local/etc*, aby zmienić ten katalog na */etc/ssh* podczas konfiguracji dodano opcję *sysconfdir*

```
$ ./configure --sysconfdir=/etc/ssh
```

```
$ make
```

```
$ make install
```

Tak zainstalowana aplikacja przechowuje pliki binarne w */usr/local/bin* oraz demona *sshd* w */usr/local/sbin*. Do uruchomienia się usługi potrzebny jest użytkownik *sshd*, w celach bezpieczeństwa nie nadano mu prawa dostępu do powłoki *shell* oraz katalog domowy ustawiono na */dev/null*. Hasło użytkownika pozostawiono puste pamiętając o tym aby przy konfiguracji SSH zablokować dostęp użytkownikom bez haseł

```
$ useradd -s /bin/false -d /dev/null sshd
```

Ponieważ większość ataków poprzez SSH skierowane jest w stronę konta *root* zablokowano do niego zdalny dostęp a utworzono dodatkowe konto *mprzeor*, które będzie służyło do zdalnego logowania

```
$ useradd mprzeor -m -c "Maciej Przeor"
```

Po poprawnym utworzeniu nowego użytkownika nadano mu silne hasło

```
$ passwd mprzeor
```

Najczęściej demon *sshd* uruchamia się z domyślną konfiguracją, zawartą w pliku */etc/ssh/ssh_config* jednak nie jest ona dość bezpieczna żeby mogła działać na serwerze dostępnym w Internecie. W konfiguracji serwera SSH ustawiono następujące opcje

Port 22 - ustawia port, na którym będzie działała aplikacja

Protocol 2 - serwer będzie wspierał tylko protokół SSH w wersji 2, który jest znacznie bezpieczniejszy od wersji 1

HostKey /etc/ssh/ssh_host_rsa_key - wskazuje plik z kluczem prywatnym rsa

HostKey /etc/ssh/ssh_host_dsa_key - wskazuje plik z kluczem prywatnym dsa

LogLevel INFO - określa poziom rejestrowanych zdarzeń, które umożliwią w przyszłości przeglądanie udanych i nieudanych logowań za pomocą SSH

LoginGraceTime 30 - ustawia czas na 30 sekund po którym nastąpi rozłączenie jeżeli użytkownik do tego czasu nie zaloguje się

PermitRootLogin no - wyłącza możliwość logowania na użytkownika root

MaxAuthTries 3 - określa maksymalną ilość nieudanych prób logowania na 2 po których nastąpi rozłączenie

PasswordAuthentication yes - włącza możliwość logowania za pomocą hasła

PermitEmptyPasswords no - uniemożliwia zalogowanie użytkownikom, którzy mają puste hasło

ClientAliveInterval 5m – ustawia czas na 5 minut po jakim serwer wyśle wiadomość z zapytaniem do klienta czy jest aktywny

ClientAliveCountMax 6 – ustawia na 6 maksymalną ilość wiadomości z zapytaniem o aktywność klienta, jeśli liczba zostanie przekroczona to serwer zrywa połączenie

Baner */var/ssh_welcome* – określa ścieżkę do pliku, który zostanie wyświetlony przed autoryzacją użytkownika

Na tak skonfigurowany serwer można się zalogować podając użytkownika oraz hasło bądź przy użyciu wcześniej wygenerowanej pary kluczy (publiczny i prywatny).

Dodatkowo utworzono powitalny plik tekstowy */var/ssh_welcome*, który będzie się wyświetlał po podłączeniu do serwera SSH

```
$ nano /var/ssh_welcome
```

```
#####  
####  WITAM NA SERWERZE PRZEOR.EU  ####  
#####
```

Listing 2.3 zawartość pliku /var/ssh_welcome

2.7 Iptables

Najnowsze źródła programu pobrano z domowej strony projektu <http://netfilter.org> po czym je rozpakowano, skonfigurowano, skompilowano i zainstalowano

```
$ wget http://netfilter.org/projects/iptables/files/iptables-1.4.2.tar.bz2
```

```
$ tar -xvjf iptables-1.4.2.tar.bz2
```

```
$ cd iptables-1.4.2
```

```
$ ./configure
```

\$ make

\$ make install

Domyślnie aplikacja instaluje się w */usr/sbin/iptables*. Wszystkie reguły dotyczące ruchu w sieci umieszczono w pliku */etc/iptables*, poniżej znajduje się zawartość pliku wraz z opisami:

- *echo "1" > /proc/sys/net/ipv4/ip_forward*

włącza przekazywanie pakietów między interfejsami dzięki czemu pakiety odebrane na interfejsie *eth0* będą mogły być przekazane na interfejs *eth1* i odwrotnie

- *iptables -F -t nat*
iptables -X -t nat
iptables -F -t filter
iptables -X -t filter

czyści wszystkie reguły w łańcuchach *nat* oraz *filter* i usuwa łańcuchy aby mieć pewność, że podczas tworzenia nowej konfiguracji w pamięci nie zostały żadne informacje po poprzednich ustawieniach

- *iptables -P INPUT DROP*
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT

blokuje cały ruch przychodzący oraz przekierowywany, dzięki czemu będzie można przepuszczać tylko pakiety spełniające poniższe warunki

- *iptables -A INPUT -i lo -j ACCEPT*

odblokowuje ruch przychodzący z interfejsu *lo*, który czasami jest wykorzystywany przez system bądź też przydaje się do testowania aplikacji

- *iptables -A INPUT -p tcp --dport 22 -j ACCEPT*
iptables -I INPUT -p tcp --dport 22 -m state --state NEW -m recent --set
iptables -I INPUT -p tcp --dport 22 -m state --state NEW -m recent --update --seconds 300 --hitcount 3 -j DROP

akceptuje pakiety z portem docelowym 22 czyli SSH, jednocześnie kontroluje ilość nowych połączeń na ten port, jeśli ich liczba przekroczy trzy w ciągu 5 minut to zaczyna odrzucać ruch na tym porcie ze źródłowego IP

- *iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT*
iptables -A INPUT -m state --state RELATED -j ACCEPT

dopuszcza połączenia wcześniej już zestawione i powiązane

- *iptables -A INPUT -p icmp -j ACCEPT*

przepuszcza pakiety protokołu icmp dzięki czemu serwer będzie odpowiadał na pingi

- *iptables -I FORWARD -p tcp -i eth0 --dport 3389 -j ACCEPT*
iptables -t nat -I PREROUTING -p tcp -i eth0 -d 0/0 --dport 3389 -j DNAT --to 192.168.1.1

przekierowuje ruch na porcie 3389 do lokalnego serwera windowsowego z adresem IP 192.168.1.1 co pozwoli na zdalne podłączanie się do serwera z Internetu

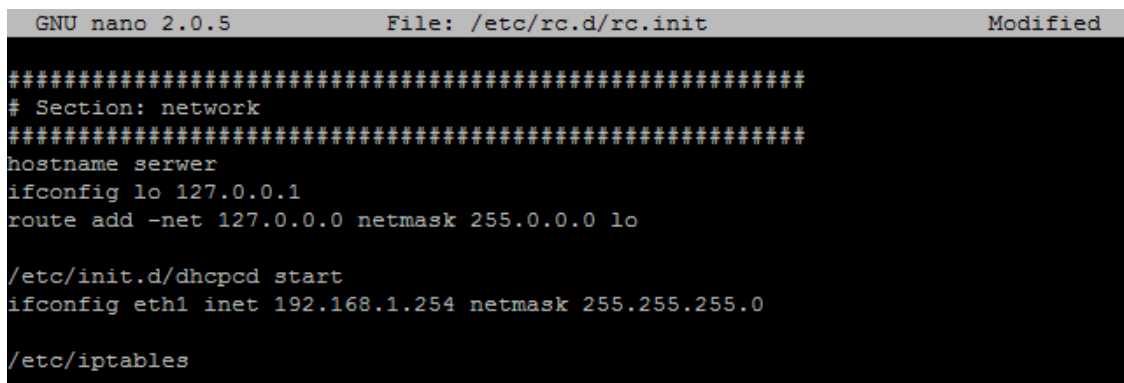
- *iptables -t nat -A POSTROUTING -s 192.168.1.0/255.255.255.0 -d 0/0 -j MASQUERADE*

włącza NAT na sieci lokalnej 192.168.1.0 tym samym umożliwiając lokalnym komputerom dostęp do Internetu przy wykorzystaniu jednego publicznego adresu IP

Aby reguły iptables były wczytywane przy każdym starcie systemu nadano uprawnienia do wykonywania dla pliku */etc/iptables* a następnie dopisano ten skrypt do pliku */etc/rc.d/rc.init*

```
$ chmod +x /etc/iptables
```

```
$ nano /etc/rc.d/rc.init
```



```
GNU nano 2.0.5           File: /etc/rc.d/rc.init           Modified
#####
# Section: network
#####
hostname server
ifconfig lo 127.0.0.1
route add -net 127.0.0.0 netmask 255.0.0.0 lo

/etc/init.d/dhcpd start
ifconfig eth1 inet 192.168.1.254 netmask 255.255.255.0

/etc/iptables
```

Rys. 2.13. sekcja sieci w pliku */etc/rc.d/rc/init*

Aby *Iptables* wraz z *Dhcpd* pełniło rolę narzędzia monitorującego sieć uruchomiono usługę *Dhcpd* w opcji debugowania a wyświetlane przez demona komunikaty o przydzielanych klientom adresach IP skierowano do skryptu */usr/sbin/dopuscip*. Skrypt ten jest odpowiedzialny za dodawanie do *iptables* reguł przepuszczających ruch od hostów z sieci lokalnej. Reguły dodawane będą do predefiniowanych łańcuchów INPUT oraz FORWARD i będą zawierały adres IP oraz MAC hosta. Pary adresów odczytywane są dla hostów

statycznych z pliku `/etc/dhcpd.conf` a dla hostów dynamicznych bezpośrednio od demona Dhcpd. Dzięki temu dostęp do serwera oraz do Internetu będą miały tylko hosty, które są statycznie wpisane w `/etc/dhcpd.conf` bądź ich adres IP został przydzielony przez Dhcp. Zastosowany mechanizm uniemożliwi przypisanie hostom statycznych adresów IP oraz ich zmianę przez użytkowników sieci a co za tym idzie realizowany monitoring ruchu będzie przedstawiał rzeczywiste wykorzystanie sieci przez danego hosta

\$ nano /usr/sbin/dopuscip

```
#!/bin/sh

Conf=`cat /etc/dhcpd.conf`;

echo "$Conf" | while read line; do
    LiniaMac=`echo $line | grep 'hardware ethernet' -c`

    if [ $LiniaMac = 1 ]; then
        Mac=`echo $line | awk '{print $3}' | awk -F';' '{print $1}'`
    fi

    LiniaIp=`echo $line | grep 'fixed-address' -c`

    if [ $LiniaIp = 1 ]; then
        Ip=`echo $line | awk '{print $2}' | awk -F';' '{print $1}'`

        iptables -A FORWARD -m mac --mac-source $Mac -s $Ip/255.255.255.255
        -d 0/0 -j ACCEPT
        iptables -A INPUT -m mac --mac-source $Mac -s $Ip/255.255.255.255 -
        d 0/0 -j ACCEPT
        iptables -A FORWARD -s 0/0 -d $Ip/255.255.255.255 -j ACCEPT
        iptables -A OUTPUT -s 0/0 -d $Ip/255.255.255.255 -j ACCEPT
    fi
done

while read line; do
    ExpLine=(${line})
    DHCPACK=`echo $ExpLine | grep DHCPACK -c`
    if [ $DHCPACK = 1 ]; then
        IP=${ExpLine[2]}
        MAC=${ExpLine[4]}

        Istnieje=`iptables -L FORWARD | awk '{print $4}' | grep $IP -c`
        if [ $Istnieje = 0 ]; then
            iptables -A FORWARD -m mac --mac-source $MAC -s
$IP/255.255.255.255 -d 0/0 -j ACCEPT
            iptables -A INPUT -m mac --mac-source $MAC -s $IP/255.255.255.255
            -d 0/0 -j ACCEPT
            iptables -A FORWARD -s 0/0 -d $IP/255.255.255.255 -j ACCEPT
            iptables -A OUTPUT -s 0/0 -d $IP/255.255.255.255 -j ACCEPT
        fi

        Czas=`date +%s`
        echo $Czas $IP $MAC >> /var/db/ipmac
    fi
done
```

Listing 2.4 zawartość pliku /usr/sbin/dopuscip

```
$ /usr/sbin/dhcpd -d eth1 2>&1 | /usr/sbin/dopuscip &
```

W celu archiwizowania wykorzystania łącza, za pomocą skryptu */usr/sbin/zapiszruch* zapisywany będzie ruch generowany przez poszczególne hosty. Skrypt odczytuje ruch dla hostów z *iptables*, jeśli dany host wygenerował ruch to jest on dopisywany wraz z datą do pliku o nazwie IP hosta w katalogu */var/ruch* (np. */var/ruch/192.168.1.210*). Plik składa się z linijek w których zapisane są trzy wartości oddzielone spacjami: czas wyrażony w sekundach, ilość bajtów wysłanych oraz ilość bajtów odebranych. Po zapisaniu danych dla wszystkich hostów skrypt zeruje liczniki *iptables*.

Przed utworzeniem skryptu stworzono katalog roboczy */var/ruch*

```
$ mkdir /var/ruch
```

```
$ nano /etc/init.d/dhcpd
```

```
Katalog='/var/ruch/'

Listing=`iptables -L FORWARD -v -x | grep 192.168.2.`;

echo "$Listing" | while read line; do
    Rodzaj=`echo $line | awk '{print $8}' | grep '192.168.2' -c`
    if [ $Rodzaj != 0 ]; then
        Ip=`echo $line | awk '{print $8}'`
        Wysyla=`echo $line | awk '{print $2}'`
    else
        Odbiera=`echo $line | awk '{print $2}'`

        if [ $Wysyla -gt 0 ]; then
            Plik=$Katalog$Ip
            Czas=`date +%s`
            Linia=`echo $Czas $Wysyla $Odbiera`
            echo $Linia >> $Plik
        fi
    fi
done

iptables -t filter -Z
```

Listing 2.5 zawartość pliku /etc/init.d/dhcpd

Po stworzeniu skryptu nadano mu prawa wykonywania i dodano go do */etc/crontab* aby wykonywał się co pięć minut

```
$ chmod +x /usr/sbin/zapiszruch
```

```
$ nano /etc/crontab
```



```
GNU nano 2.0.5      File: /etc/crontab      Modified
* * * * * root /usr/sbin/gwping
*/5 * * * * root /usr/sbin/zapiszruch
```

Rys. 2.14. zrzut ekranu ilustrujący zawartość pliku */etc/crontab*

2.8 Lighttpd + PHP

Kolejność instalacji programów Lighttpd oraz PHP jest dowolna gdyż działają one niezależnie od siebie. Jako pierwszą na serwerze zainstalowano bibliotekę *libxml*, która jest niezbędna do prawidłowego skompilowania źródeł PHP. Biblioteka jest darmowa i można ją pobrać z oficjalnej strony projektu <http://xmlsoft.org/>

```
$ wget ftp://xmlsoft.org/libxml2/libxml2-2.7.3.tar.gz
$ ./configure
$ make
$ make install
```

Po pomyślnym wgraniu biblioteki, zainstalowano PHP w trybie FastCGI

```
$ wget http://pl.php.net/get/php-5.2.10.tar.gz/from/this/mirror
$ tar xzvf php-5.2.10.tar.gz
$ cd php-5.2.10
$ ./configure --enable-fastcgi
$ make
$ make install
```

Obszerny plik konfiguracyjny PHP znajduje się w */etc/php.ini* jednak nie wymaga on żadnych zmian, ponieważ standardowe ustawienia spiszą się doskonale w przypadku realizowanego serwera.

Po prawidłowym zainstalowaniu PHP przystąpiono do instalacji Lighttpd, którego źródła można pobrać z oficjalnej strony projektu <http://lighttpd.net/>. W celu uniknięcia instalacji dużej ilości bibliotek, których wymaga PCRE całkowicie zrezygnowano z tego modułu

```
$ wget http://www.lighttpd.net/download/lighttpd-1.4.22.tar.gz
$ tar xzvf lighttpd-1.4.22.tar.gz
$ cd lighttpd-1.4.22
```

```
$ ./configure --without-pcre
```

```
$ make
```

```
$ make install
```

Następnie z katalogu *src* skopiowano program *spawn-fcgi*, który odpowiedzialny będzie za uruchomienie serwera fcgi

```
$ cp src/spawn-fcgi /usr/bin/spawn-fcgi
```

Do prawidłowego działania aplikacji Lighttpd potrzebny jest użytkownik *lighttpd*, katalog przechowujący pliki dziennikowe oraz katalog zawierający pliki ze stronami *html*, którego właścicielem będzie użytkownik *lighttpd*

```
$ useradd -d /var/www/html -s /bin/false lighttpd
```

```
$ mkdir /var/log/lighttpd
```

```
$ mkdir /var/www
```

```
$ chown lighttpd:lighttpd /var/www
```

Całą konfigurację programu umieszczono w pliku */etc/lighttpd.conf*. Zdefiniowano moduły, jakie mają być załadowane podczas uruchamiania programu: pierwszy do zapisywania plików dziennikowych, drugi do obsługi serwera fcgi, czyli de facto PHP

```
server.modules = ("mod_accesslog", "mod_fastcgi")
```

wskazano, że pliki z rozszerzeniem *php* mają zostać najpierw obsługane przez serwer fastcgi a dopiero później wysłane do klienta

```
fastcgi.server = (".php" => (("bin-path" => "/usr/local/bin/php-cgi", "socket" => "/tmp/php.socket")))
```

przypisano port, na jakim działa usługa

```
server.port = 80
```

zdefiniowano użytkownika i grupę, których aplikacja będzie używała do operacji na plikach

```
server.username = "lighttpd"
```

```
server.groupname = "lighttpd"
```

wskazano plik przechowujący *pid* demona Lighttpd

```
server.pid-file = "/var/run/lighttpd.pid"
```

określono katalog zawierający pliki ze stronami html, nazwy plików indeksów oraz wyłączono możliwość przeglądania katalogu, jeśli nie zawiera on pliku z indeksem

```
server.document-root = "/var/www/"  
index-file.names = ("index.php", "index.html")  
dir-listing.activate = "disable"
```

zdefiniowano również ścieżki do plików dziennikowych dostępu oraz błędów

```
accesslog.filename = "/var/log/lighttpd/access.log"  
server.errorlog = "/var/log/lighttpd/error.log"
```

2.9 Samba

Najnowsze źródła Samby pobrano z oficjalnej strony projektu <http://samba.org/>, po czym je zainstalowano

```
$ wget http://samba.org/samba/ftp/stable/samba-3.0.9.tar.gz  
$ tar -zxvf samba-3.0.9.tar.gz  
$ cd samba-3.0.9/source/  
$ ./configure --libdir=/usr/lib  
$ make  
$ make install
```

Domyślnie Samba instaluje się */usr/local/samba* a pliki wykonywalne programu */usr/local/samba/sbin/*. Do prawidłowego działania serwera plików potrzeby jest użytkownik w systemie, z którego Samba będzie korzystała podczas operacji na plikach. Najczęściej do tych celów wykorzystywany jest użytkownik *nobody* należący do grupy *no group*

```
$ useradd -d /dev/null -s /bin/false nobody
```

Konfiguracja Samby przechowywana jest w pliku tekstowym *smb.conf*, znajdującym się najczęściej w katalogu */usr/local/samba/lib/*. Przed rozpoczęciem konfiguracji stworzono dowiązanie symboliczne do tego pliku w katalogu */etc/*.

```
$ ln -s /usr/local/samba/lib/smb.conf /etc/smb.conf
```

Plik konfiguracyjny podzielony jest na sekcje, których nazwy zawarte są w kwadratowych nawiasach. Global to najważniejsza sekcja, w której zawarte są ustawienia mające wpływ na działanie całego serwera Samby. Jest to jedyna sekcja wymagana do prawidłowego działania oprogramowania. Określono w niej nazwę grupy roboczej, nazwę

netbios, opis serwera, sposób udostępniania plików umożliwiający przeglądanie zasobów bez logowania, dostęp tylko dla komputerów z sieci lokalnej, nie instalowanie drukarek

[global]

workgroup = lokalna

netbios name = serwer

server string = serwer samba

security = share

hosts allow = 192.168.1.

load printers = no

oraz obsługę logowania zdarzeń na najmniejszym poziomie w pliku, którego wielkość nie będzie przekraczała 1024 KB

log file = /var/log/samba

log level = 1

max log size = 1024

Na serwerze udostępniono dwa zasoby: katalog *temp* z prawami do zapisu, służący użytkownikom do wymiany dokumentów między sobą

[temp]

comment = katalog roboczy

path = /var/temp

create mask = 0777

directory mask = 0777

browseable = yes

writable = yes

public = yes

oraz katalog *filmy* umożliwiający jedynie pobieranie zbiorów zgromadzonych na serwerze

[filmy]

comment = ciekawe filmy

path = /var/filmy

browseable = yes

read only = yes

public = yes

Po zakończeniu konfiguracji przetestowano plik */etc/smb.conf* aby upewnić się, że nie zawiera żadnych błędów składniowych

```
testparm /etc/smb.conf
```

oraz stworzono katalogi do przechowywania danych, które zostały wymienione w pliku konfiguracyjnym

```
$ mkdir /var/temp
```

```
$ chmod 777 /var/temp
```

```
$ mkdir /var/filmy
```

Plik */etc/smb.conf* jest regularnie sprawdzany przez wszystkie demony Samby i należy maksymalnie ograniczyć jego wielkość, dlatego zostały z nim pominięte wszelkiego rodzaju komentarze.

2.10 Poptop – PPTP Server

Najnowsze źródła demona PPTP pobrano z oficjalnej strony projektu <http://poptop.org/>, po czym je skompilowano i zainstalowano

```
$ wget http://mesh.dl.sourceforge.net/sourceforge/poptop/pptpd-1.3.4.tar.gz
```

```
$ tar -zxvf pptpd-1.3.4.tar.gz
```

```
$ cd pptpd-1.3.4
```

```
$ ./configure
```

```
$ make
```

```
$ make install
```

Domyślnie aplikacja korzysta z głównego pliku konfiguracyjnego */etc/pptpd.conf* oraz z plików pomocniczych znajdujących się w katalogu */etc/ppp/*. W pierwszym pliku ustawiono następujące opcje

ścieżkę do pliku z zaawansowanymi opcjami PPTP

```
option /etc/ppp/options.pptpd
```

lokalne IP serwera, które będzie bramą dla podłączających się hostów

```
localip 192.168.2.254
```

pulę zdalnych IP, które będą przydzielane podłączającym się hostom

```
remoteip 192.168.2.1-253
```

Nastomiast w pliku */etc/ppp/options.pptpd* umieszczono konfiguracje dotyczącą autoryzacji i szyfrowania

włączono autoryzację dla połączeń

```
auth
```

odrzucono połączenia autoryzowane za pomocą PAP, CHAP oraz MS-CHAPv1 w wersji pierwszej

```
refuse-pap
```

```
refuse-chap
```

```
refuse-mschap
```

wymuszono logowanie przy pomocy MS-CHAPv2 oraz szyfrowanie MPPE-128

```
require-mschap-v2
```

```
require-mppe-128
```

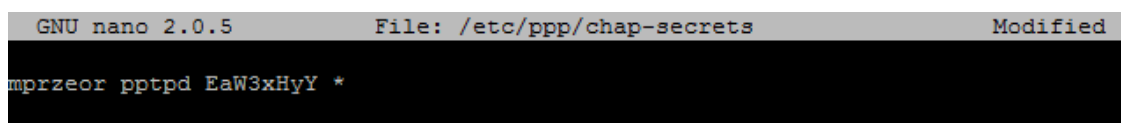
wskazano podłączającym się hostom adresy serwerów DNS

```
ms-dns 194.204.159.1
```

```
ms-dns 194.204.152.34
```

W pliku */etc/ppp/chap-secrets* umieszczono dane potrzebne do zalogowania się na serwer: nazwę użytkownika, nazwę demona odpowiadającego za połączenie *ppp*, hasło użytkownika oraz adresy IP z przypisanej puli, które mogą być dzierżawione temu klientowi

```
$ nano /etc/ppp/chap-secrets
```



```
GNU nano 2.0.5      File: /etc/ppp/chap-secrets      Modified
mprzeor pptpd EaW3xHyY *
```

*Rys. 2.15. dane potrzebne do logowania przechowywane
w plik /etc/ppp/chap-secrets*

Plik zawiera w sobie niezaszyfrowane hasło, więc dla bezpieczeństwa ograniczono mu prawa dostępu tylko z poziomu konta *root*

```
$ chmod 700 /etc/ppp/chap-secrets
```

W celu umożliwienia podłączenia się z serwerem w pliku */etc/iptables* dodano regułę, która przepuszcza ruch z interfejsu *eth0* na porcie *1723*, na którym domyślnie działa *pptpd*

```
iptables -A INPUT -i eth0 -p tcp --dport 1723 -j ACCEPT
```

Ponadto umieszczono reguły zapewniające klientom dostęp do serwera i Internetu już po ustanowieniu połączenia *ppp*

```
iptables -A FORWARD -i ppp+ -s 192.168.2.0/255.255.255.0 -d 0/0 -j ACCEPT
```

```
iptables -A FORWARD -s 0/0 -d 192.168.2.0/255.255.255.0 -j ACCEPT
```

```
iptables -A INPUT -i ppp+ -s 192.168.2.0/255.255.255.0 -d 0/0 -j ACCEPT
```

```
iptables -A INPUT -s 0/0 -d 192.168.2.0/255.255.255.0 -j ACCEPT
```

2.11 Zarządzanie usługami

Przeszukiwanie listy procesów w celu wyłączenia usługi lub pamiętanie składni poleceń uruchamiających dane programy jest niewygodne i niewydajne, dlatego też w realizowanym systemie stworzono skrypty do zarządzania usługami. Każda z wyżej opisanych usług ma skrypt startowy w katalogu */etc/init.d/* odpowiedzialny za zarządzanie usługą. Poniżej przedstawiono przykładowy skrypt umożliwiający łatwe włączenie, wyłączenie lub zrestartowanie usługi *sshd*

```
$ nano /etc/init.d/sshd
```

```
#!/bin/sh

PlikPid='/var/run/sshd.pid'

Start () {
    if [ -a $PlikPid ]; then
        Pid=`cat $PlikPid -s`
        if [ `ps -A | grep -c $Pid` != 1 ]; then
            echo "Usuwanie $PlikPid"
            rm $PlikPid
            echo "Uruchamiam sshd"
            /usr/local/sbin/sshd
        else
            echo "Sshd jest już uruchomione"
        fi
    else
        echo "Uruchamiam sshd"
        /usr/local/sbin/sshd
    fi
}

Stop() {
    if [ -a $PlikPid ]; then
        echo "Wyłączam sshd"
        kill `cat $PlikPid`
    else
        echo "Sshd nie jest uruchomione"
    fi
}

Restart() {
    Stop
    sleep 2
    Start
}

case "$1" in
    'start')
        Start
        ;;
    'stop')
        Stop
        ;;
    'restart')
        Restart
        ;;
    *)
        echo "Sposób użycia: $0 {start | stop | restart}"
        ;;
esac
```

Listing 2.6 zawartość pliku /etc/init.d/sshd

Aby usługi uruchamiały się wraz ze startem systemu do pliku */etc/rc.d/rc.init* dodano sekcję *programs*, w której zawarto ścieżki skryptów startowych z parametrem *start*


```

GNU nano 2.0.5      File: /etc/rc.d/rc.init

#####
# Section: programs
#####

/etc/init.d/cron start
/etc/init.d/sshd start
/etc/init.d/dhcpd start
/etc/init.d/dhcpd start
/etc/init.d/php-fcgi start
/etc/init.d/lighttpd start
/etc/init.d/smb start
/etc/init.d/pptpd start

```

Rys. 2.16. sekcja programów w pliku /etc/rc.d/rc.init

3. Zarządzanie serwerem i testy systemu

3.1 Zarządzanie serwerem

Wykorzystując język PHP napisałem skrypty umożliwiające wygodne monitorowanie ruchu oraz zarządzanie serwerem przez www. Gotowe skrypty umieściłem w katalogu /var/www. Kody źródłowe skryptów znajdują się pod koniec rozdziału. Strona www udostępniona jest tylko dla hostów znajdujących się w sieci lokalnej. Po wpisaniu w przeglądarce adresu IP serwera (192.168.1.254) lub nazwy NetBIOS (serwer) wyświetlany jest monitoring ruchu sieci 192.168.1.0

17-08-2009 17:43:07

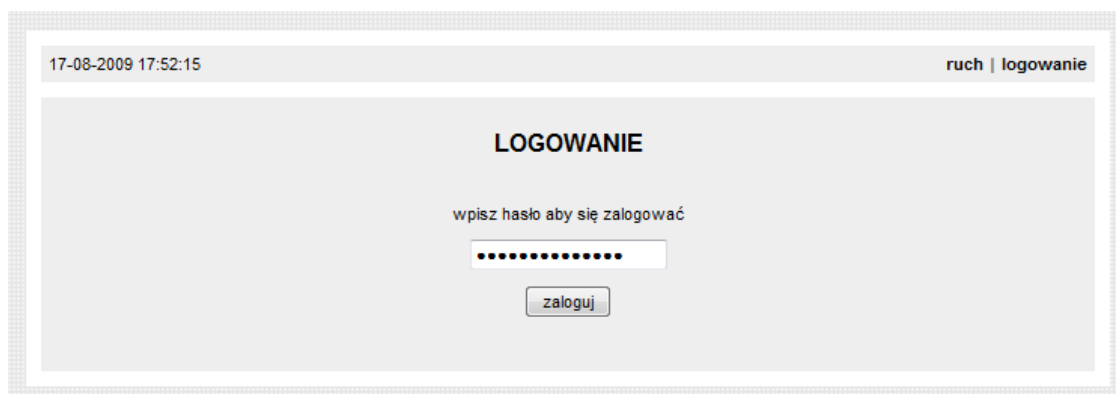
ruch | logowanie

RUCH Z SIECI 192.168.1.0 DO INERNETU							
nazwa	godzina		dzień		miesiąc		aktywny
	wysłane	odebrane	wysłane	odebrane	wysłane	odebrane	
192.168.1.1	2.25MB 656.32B/s	50.78MB 14.44KB/s	475.66MB 5.64KB/s	433.94MB 5.14KB/s	9.8GB 3.96KB/s	12.44GB 5.03KB/s	17:40:00
192.168.1.10	0B 0B/s	0B 0B/s	0B 0B/s	0B 0B/s	40.61MB 16.43B/s	2.05GB 847.71B/s	21:15:00 14-08-2009
192.168.1.112	120.67KB 34.32B/s	3.31MB 963.2B/s	1.1MB 13.32B/s	8.46MB 102.71B/s	12.68MB 5.13B/s	78.39MB 31.71B/s	17:40:00
192.168.1.186	0B 0B/s	0B 0B/s	22.17MB 269.04B/s	32.96MB 400.06B/s	51.18MB 20.7B/s	368.96MB 149.26B/s	12:05:00
192.168.1.200	136B 0.04B/s	160B 0.04B/s	1003.18KB 11.89B/s	5.16MB 62.58B/s	12.13MB 4.91B/s	52.61MB 21.28B/s	16:45:00
192.168.1.201	0B 0B/s	0B 0B/s	0B 0B/s	0B 0B/s	13.02MB 5.27B/s	224.94MB 91B/s	13:30:00 14-08-2009
suma	2.37MB 690.68B/s	54.09MB 15.39KB/s	499.91MB 5.92KB/s	480.53MB 5.7KB/s	9.93GB 4.02KB/s	15.19GB 6.15KB/s	-

uwaga, dane odświeżane są co pięć minut

Rys. 3.1. widok strony monitoringu ruchu - <http://192.168.1.254/>

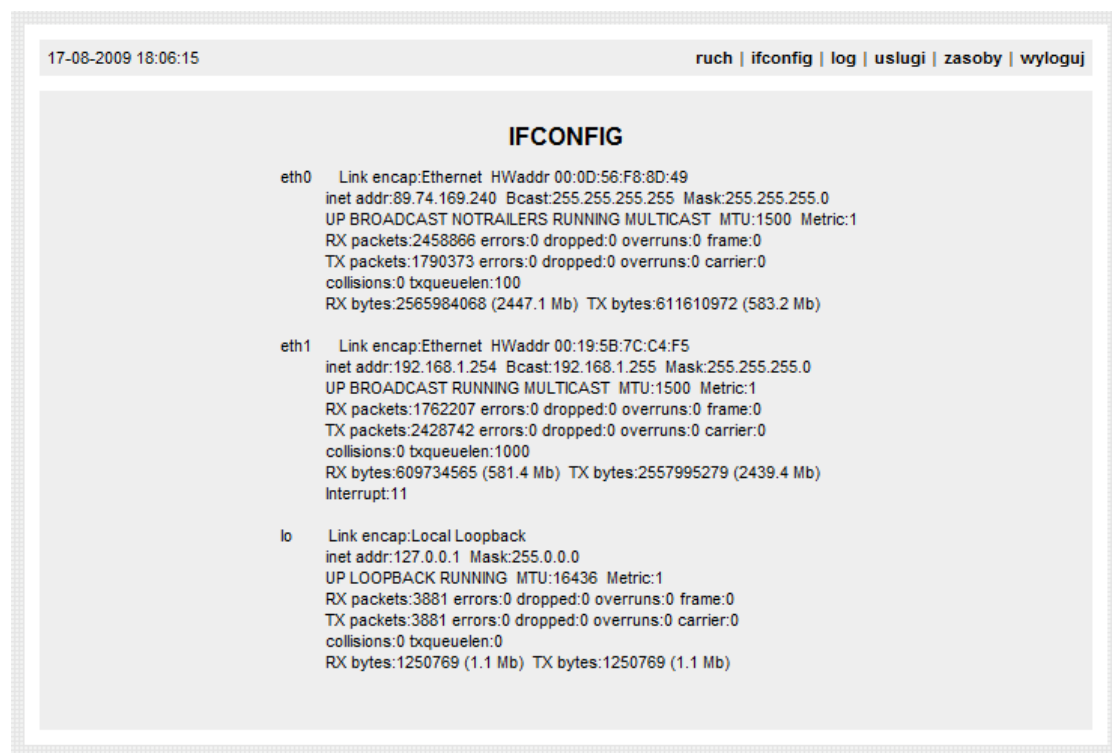
Na każdej stronie umieszczone jest menu umożliwiające przełączanie się między podstronami. Dla niezalogowanych użytkowników udostępniony jest tylko link do przejścia na stronę logowania



The screenshot shows a web interface for logging in. At the top left, the date and time are '17-08-2009 17:52:15'. At the top right, there is a menu with 'ruch' and 'logowanie'. The main heading is 'LOGOWANIE'. Below it, the text 'wpisz hasło aby się zalogować' is displayed. There is a password input field represented by a series of dots. Below the input field is a button labeled 'zaloguj'.

Rys. 3.2. widok strony logowania - <http://192.168.1.254/?s=logowanie>

Po zalogowaniu pojawiają się dodatkowe opcje umożliwiające kontrolowanie i zarządzanie serwerem. Na podstronie *ifconfig* wyświetlane są wszystkie dostępne w systemie interfejsy sieciowe



The screenshot shows the 'IFCONFIG' page. At the top left, the date and time are '17-08-2009 18:06:15'. At the top right, there is a menu with 'ruch', 'ifconfig', 'log', 'uslugi', 'zasoby', and 'wyloguj'. The main heading is 'IFCONFIG'. Below it, the details for three network interfaces are listed:

- eth0**: Link encap:Ethernet HWaddr 00:0D:56:F8:8D:49
inet addr:89.74.169.240 Bcast:255.255.255.255 Mask:255.255.255.0
UP BROADCAST NOTRAILERS RUNNING MULTICAST MTU:1500 Metric:1
RX packets:2458866 errors:0 dropped:0 overruns:0 frame:0
TX packets:1790373 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100
RX bytes:2565984068 (2447.1 Mb) TX bytes:611610972 (583.2 Mb)
- eth1**: Link encap:Ethernet HWaddr 00:19:5B:7C:C4:F5
inet addr:192.168.1.254 Bcast:192.168.1.255 Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:1762207 errors:0 dropped:0 overruns:0 frame:0
TX packets:2428742 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:609734565 (581.4 Mb) TX bytes:2557995279 (2439.4 Mb)
Interrupt:11
- lo**: Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:3881 errors:0 dropped:0 overruns:0 frame:0
TX packets:3881 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:1250769 (1.1 Mb) TX bytes:1250769 (1.1 Mb)

Rys. 3.3. widok strony z konfiguracją interfejsów - <http://192.168.1.254/?s=ifconfig>

Podstrona *log* pozwala obejrzeć ostatnie wpisy w pliku dziennikowym */etc/log/messages* oraz datę ostatnich logowań wszystkich użytkowników

17-08-2009 18:50:03

ruch | ifconfig | log | usługi | zasoby | wyloguj

OSTATNIE 20 ZDARZEŃ W SYSTEMIE

```
Aug 17 16:21:53 serwer dhcpd: DHCPACK on 192.168.1.1 to 00:1d:60:d8:13:08 via eth1
Aug 17 16:22:16 serwer dhcpd: DHCPDISCOVER from 00:1d:60:d8:13:08 via eth1
Aug 17 16:22:16 serwer dhcpd: DHCPPOFFER on 192.168.1.1 to 00:1d:60:d8:13:08 via eth1
Aug 17 16:22:16 serwer dhcpd: DHCPREQUEST for 192.168.1.1 (192.168.1.254) from 00:1d:60:d8:13:08 via eth1
Aug 17 16:22:16 serwer dhcpd: DHCPACK on 192.168.1.1 to 00:1d:60:d8:13:08 via eth1
Aug 17 16:39:12 serwer dhcpd: DHCPINFORM from 192.168.1.200 via eth1
Aug 17 16:39:12 serwer dhcpd: DHCPACK to 192.168.1.200 (00:1f:3b:7d:6e:61) via eth1
Aug 17 17:08:00 serwer -- MARK --
Aug 17 17:28:00 serwer -- MARK --
Aug 17 17:31:46 serwer sshd[15714]: error: Could not get shadow information for mprzeor
Aug 17 17:31:46 serwer sshd[15714]: Accepted password for mprzeor from 192.168.1.1 port 52038 ssh2
Aug 17 17:48:00 serwer -- MARK --
Aug 17 17:59:42 serwer sshd[17031]: error: Could not get shadow information for root
Aug 17 17:59:42 serwer sshd[17031]: Failed password for root from 61.189.0.179 port 44666 ssh2
Aug 17 17:59:42 serwer sshd[17031]: Excess permission or bad ownership on file /var/log/btmp
Aug 17 17:59:52 serwer sshd[17035]: error: Could not get shadow information for root
Aug 17 17:59:52 serwer sshd[17035]: Failed password for root from 61.189.0.179 port 45026 ssh2
Aug 17 17:59:52 serwer sshd[17035]: Excess permission or bad ownership on file /var/log/btmp
Aug 17 18:28:00 serwer -- MARK --
Aug 17 18:48:00 serwer -- MARK --
```

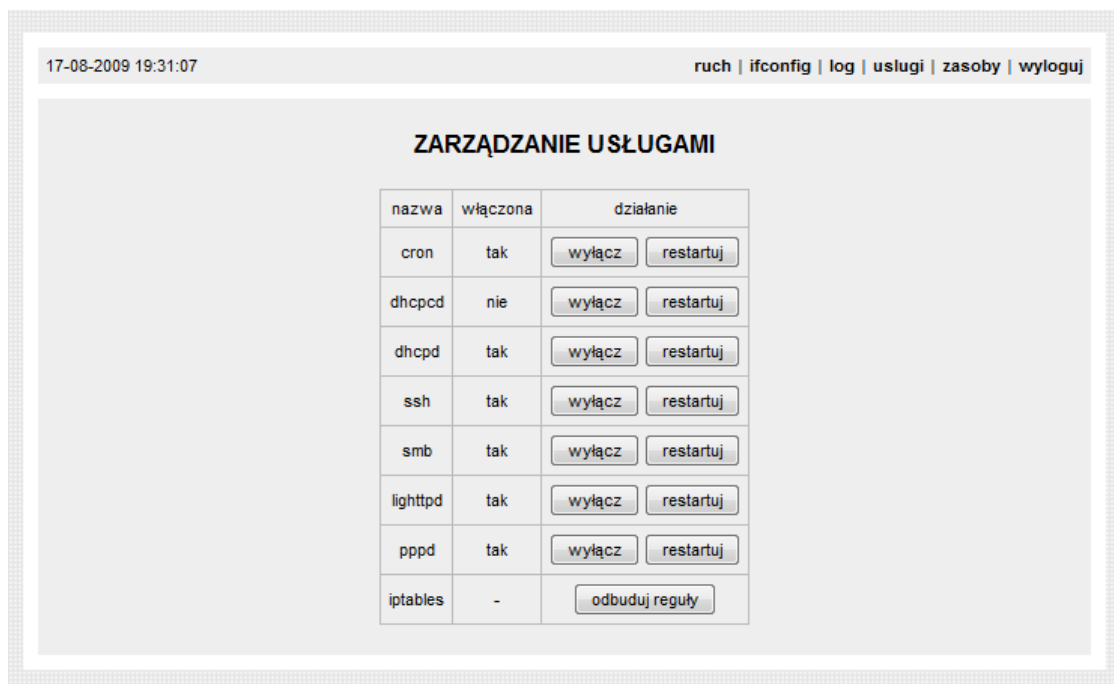
LASTLOG

użytkownik	port	skąd	data
root	tty1	-	Wed Jun 17 00:03:59 +0200 2009
nobody	-	-	**Never logged in**
sshd	-	-	**Never logged in**
mprzeor	pts/2	192.168.1.1	Mon Aug 17 17:31:46 +0200 2009
lighttpd	-	-	**Never logged in**

Rys. 3.4. - widok strony z podgląd ostatnich zdarzeń

w systemie - <http://192.168.1.254/?s=log>

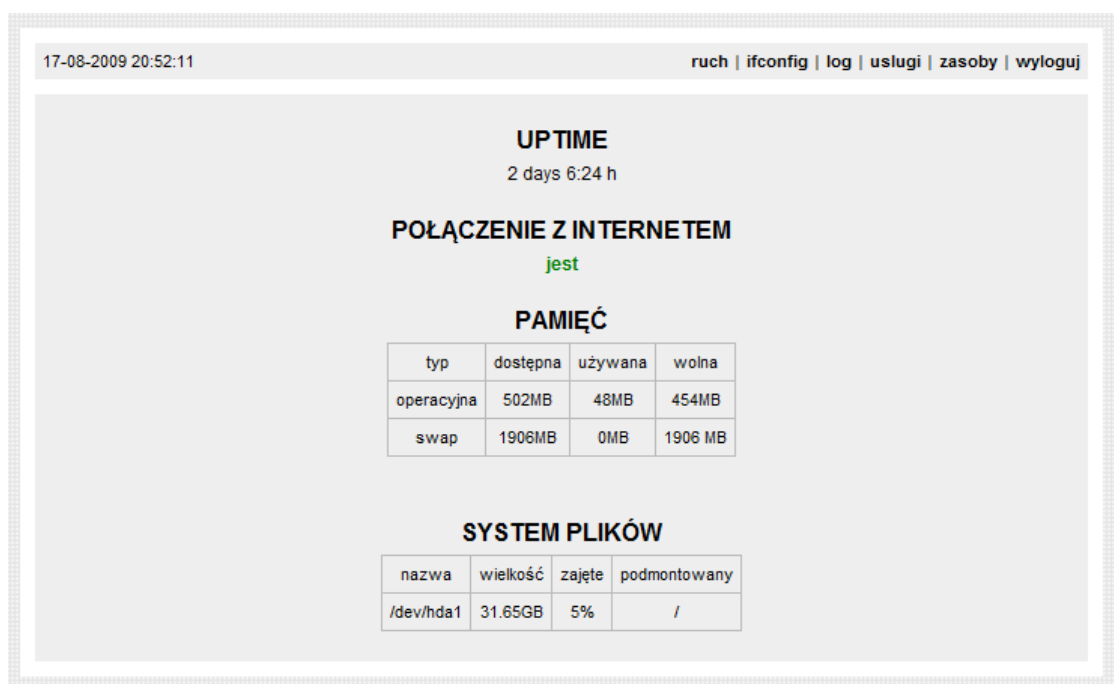
W zakładce *usługi* udostępniony został formularz umożliwiający włączanie, wyłączanie i restartowanie usług oraz odbudowywanie reguł Iptables



Rys. 3.5. zarządzanie usługami - <http://192.168.1.254/?s=uslugi>

Podstrona *zasoby* zawiera przydatne informacje o stanie serwera:

- czas jaki upłynął od uruchomienia systemu
- czy serwer połączony jest z Internetem
- ilość dostępnej pamięci operacyjnej oraz swap
- ilość wolnego miejsca na dysku



Rys. 3.6. widok strony z danymi dotyczącymi zasobów serwera - <http://192.168.1.254/?s=zasoby>

Skrypt *index.php* odpowiedzialny jest za obsługę logowania, wyświetlanie nagłówka, menu strony, stopki oraz podstron według parametrów przekazanych w pasku adresowym przeglądarki

```
<?php
    require('funkcje.php');
    $zalogowany = logowanie();
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <title>zarządzaj serwerem</title>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
        <link rel="stylesheet" href="style.css" type="text/css">
    </head>
    <body>
        <div id="main">
            <div id="naglowek">
                <div id="czas"><?php echo date('d-m-Y H:i:s') ?></div>
                <div id="menu">
                    <a href="?s=ruch">ruch</a> |
                    <?php
                        if($zalogowany) {
                            ?>
                            <a href="?s=ifconfig">ifconfig</a> |
                            <a href="?s=log">log</a> |
                            <a href="?s=uslugi">uslugi</a> |
                            <a href="?s=zasoby">zasoby</a> |
                            <a href="?wyloguj=1">wyloguj</a>
                        <?php
                            } else {
                                echo '<a href="?s=logowanie">logowanie</a>';
                            }
                        ?>
                    </div>
                </div>
            <?php
                $strona = 'ruch.php';
                if(isset($_GET['s']) AND !empty($_GET['s']))
                    $strona = $_GET['s'] . '.php';

                include $strona;
            ?>
        </body>
    </html>
```

Listing 3.1 zawartość pliku /var/www/index.php

Skrypt *ruch.php* zlicza ruch sieciowy na podstawie danych z katalogu */var/ruch* i wyświetla wynik w tabeli

```

<h1>RUCH Z SIECI 192.168.1.0 DO INERNETU</h1>

<table>
  <tr>
    <td rowspan="2">nazwa</td><td colspan="2">godzina</td>
    <td colspan="2">dzień</td><td colspan="2">miesiąc</td>
    <td rowspan="2">aktywny</td>
  </tr>
  <tr>
    <td>wysłane</td><td>odebrane</td>
    <td>wysłane</td><td>odebrane</td>
    <td>wysłane</td><td>odebrane</td>
  </tr>
</table>

<?php
$godzina = date('U') - 60*60;
$dzien = date('U') - 60*60*24;
$miesiac = date('U') - 60*60*24*30;

foreach(glob('/var/ruch/*') as $plik) {
    $mod = filemtime($plik);
    if(date('dmY') == date('dmY', $mod))
        $mod = date('H:i:s', $mod);
    else
        $mod = date('H:i:s', $mod) . '<br>' . date('d-m-Y', $mod);

    $suchwyt = fopen ($plik, "r");
    if ($suchwyt) {
        while (!feof($suchwyt)) {
            $buffer = fgets($suchwyt);
            $exp = explode(' ', $buffer);
            if($exp[0] > $godzina) {
                $wyslaneGodzina += $exp[1];
                $odebraneGodzina += $exp[2];
            }
            if($exp[0] > $dzien) {
                $wyslaneDzien += $exp[1];
                $odebraneDzien += $exp[2];
            }
            if($exp[0] > $miesiac) {
                $wyslaneMiesiac += $exp[1];
                $odebraneMiesiac += $exp[2];
            }
        }
        fclose ($suchwyt);
    }

    echo '<tr>';
    echo '<td>' . basename($plik) . '</td>';
    echo '<td>' . zaokr($wyslaneGodzina) . 'B<br>'
        . zaokr($wyslaneGodzina / (60*60)) . 'B/s</td>';
    echo '<td>' . zaokr($odebraneGodzina) . 'B<br>'
        . zaokr($odebraneGodzina / (60*60)) . 'B/s</td>';
    echo '<td>' . zaokr($wyslaneDzien) . 'B<br>'
        . zaokr($wyslaneDzien / (60*60*24)) . 'B/s</td>';
    echo '<td>' . zaokr($odebraneDzien) . 'B<br>'
        . zaokr($odebraneDzien / (60*60*24)) . 'B/s</td>';
    echo '<td>' . zaokr($wyslaneMiesiac) . 'B<br>'
        . zaokr($wyslaneMiesiac / (60*60*24*30)) . 'B/s</td>';
    echo '<td>' . zaokr($odebraneMiesiac) . 'B<br>'
        . zaokr($odebraneMiesiac / (60*60*24*30)) . 'B/s</td>';
    echo '<td>' . $mod . '</td>';
    echo '</tr>';

    $wyslaneGodzinaSuma += $wyslaneGodzina;
    $odebraneGodzinaSuma += $odebraneGodzina;

    $wyslaneDzienSuma += $wyslaneDzien;
    $odebraneDzienSuma += $odebraneDzien;

    $wyslaneMiesiacSuma += $wyslaneMiesiac;
    $odebraneMiesiacSuma += $odebraneMiesiac;
}

```

```

echo '<tr>';
echo '<td>suma</td>';
echo '<td>' . zaokr($wyslaneGodzinaSuma) . 'B<br>'
    . zaokr($wyslaneGodzinaSuma / (60*60)) . 'B/s</td>';
echo '<td>' . zaokr($odebraneGodzinaSuma) . 'B<br>'
    . zaokr($odebraneGodzinaSuma / (60*60)) . 'B/s</td>';
echo '<td>' . zaokr($wyslaneDzienSuma) . 'B<br>'
    . zaokr($wyslaneDzienSuma / (60*60*24)) . 'B/s</td>';
echo '<td>' . zaokr($odebraneDzienSuma) . 'B<br>'
    . zaokr($odebraneDzienSuma / (60*60*24)) . 'B/s</td>';
echo '<td>' . zaokr($wyslaneMiesiacSuma) . 'B<br>'
    . zaokr($wyslaneMiesiacSuma / (60*60*24*30)) . 'B/s</td>';
echo '<td>' . zaokr($odebraneMiesiacSuma) . 'B<br>'
    . zaokr($odebraneMiesiacSuma / (60*60*24*30)) . 'B/s</td>';
echo '<td> - </td>';
echo '</tr>';

?>

</table>

<h2>uwaga, dane odświeżane są co pięć minut</h2>

```

Listing 3.2 zawartość pliku /var/www/ruch.php

Skrypt *logowanie.php* zawiera jedynie formularz logowania umożliwiający wprowadzenie hasła

```

<h1>LOGOWANIE</h1>

<?php
if($zalogowany) {
    include('main.php');
} else {
?>
    <form id="logowanie" action="" method="post">
        <?php
        if(isset($_POST['haslo']))
            echo 'podałeś nieprawidłowe hasło,';

        ?>
        <p>wpisz hasło aby się zalogować</p>
        <input type="password" name="haslo">
        <br>
        <input type="submit" value="zaloguj">
    </form>
<?php
}

```

Listing 3.3 zawartość pliku /var/www/logowanie.php

Skrypt *ifconfig.php* wykonuje systemowe polecenie *ifconfig* i zwraca wynik w uporządkowanej formie

```
<h1>IFCONFIG</h1>

<?php
if(!zalogowany)
    exit('musisz być zalogowany');
?>

<pre id="ifconfig">

<?php
echo shell_exec('ifconfig');
?>

</pre>
```

Listing 3.4 zawartość pliku /var/www/ifconfig.php

Skrypt *uslugi.php* składa się z szeregu formularzy, które umożliwiają zarządzanie usługami serwera przy pomocy skryptów znajdujących się w katalogu */etc/ini.d/*


```

<?php
if(!zalogowany)
    exit('musisz być zalogowany');

if(isset($_POST['iptables'])) {
    system('/etc/iptables');
    echo '<p>odbudowano reguły iptables</p>';
    sleep(2);
}

if(isset($_POST['włącz'])) {
    system('/etc/init.d/' . $_POST['włącz'] . ' start >/dev/null 2>&1 &');
    echo '<p>włączono usługę ' . $_POST['włącz'] . '</p>';
    sleep(2);
}

if(isset($_POST['restartuj'])) {
    system('/etc/init.d/' . $_POST['restartuj'] . ' restart >/dev/null 2>&1 &');
    echo '<p>zrestartowano usługę ' . $_POST['restartuj'] . '</p>';
    sleep(2);
}

if(isset($_POST['wyłącz'])) {
    system('/etc/init.d/' . $_POST['wyłącz'] . ' stop >/dev/null 2>&1 &');
    echo '<p>wyłączono usługę ' . $_POST['wyłącz'] . '</p>';
    sleep(2);
}

$uslugi = array('cron', 'dhcpcd', 'dhcpcd', 'ssh', 'smb', 'lighttpd', 'pppd');
?>

<h1>ZARZĄDZANIE USŁUGAMI</h2>

<table>
<tr><td>nazwa</td><td>włączona</td><td>działanie</td></tr>
<?php
foreach($uslugi as $usluga) {
    echo '<tr><td>' . $usluga . '</td>';
    echo '<td>';
        $włączona = (int) shell_exec('ps -A | grep -c ' . $usluga);
        if($włączona) {
            echo 'tak';
        } else {
            echo 'nie';
        }
    echo '</td><td>';
        if($włączona) {
            echo '<form action="" method="post" class="uslugi">
                <input type="hidden" name="wyłącz" value="" . $usluga . "">
                <input type="submit" value="wyłącz"></form>
                <form action="" method="post" class="uslugi">
                <input type="hidden" name="restartuj" value="" . $usluga . "">
                <input type="submit" value="restartuj"></form>';
        } else {
            echo '<form action="" method="post">
                <input type="hidden" name="włącz" value="" . $usluga . "">
                <input type="submit" value="włącz"></form>';
        }
    echo '</td>';
    echo '</tr>';
}
?>
<tr>
<td>iptables</td><td>-</td>
<td>
        <form action="" method="post">
            <input type="hidden" name="iptables" value="" . $usluga . "">
            <input type="submit" value="odbuduj reguły">
        </form>
    </td>
</tr>
</table>

```

Listing 3.5 zawartość pliku /var/www/uslugi.php

Skrypt *zasoby.php* na podstawie poleceń systemowych *uptime*, *free* oraz *df* wyświetla przydatne informacje na temat stanu serwera

```
<h1>UPTIME</h1>
<?php

$expUptime = explode('up ', shell_exec('uptime'));
$expUptime = explode(' ', $expUptime[1]);
$uptime = $expUptime[0];

echo '<p class="zasoby">' . $uptime . ' h</p>';

?>

<h1>POŁĄCZENIE Z INTERNETEM</h1>
<?php

$ping = (int) shell_exec('cat /var/cache/badgwping');
if(!$ping)
    echo '<p class="zasoby pingok">jest</p>';
else
    echo '<p class="zasoby pingbrak">brak</p>';

?>

<h1>PAMIĘĆ</h1>
<table class="zasoby">
    <tr><td>typ</td><td>dostępna</td><td>używana</td><td>wolna</td></tr>

<?php

$mem = ereg_replace (' +', ' ', (shell_exec('free -m | grep Mem')));
$expMem = explode(' ', $mem);

$swap = ereg_replace (' +', ' ', (shell_exec('free -m | grep Swap')));
$expSwap = explode(' ', $swap);

echo '<tr><td>operacyjna</td><td>' . $expMem[1] . 'MB</td><td>' . $expMem[2] .
'MB</td><td>' . $expMem[3] . 'MB</td></tr>';
echo '<tr><td>swap</td><td>' . $expSwap[1] . 'MB</td><td>' . $expSwap[2] . 'MB</td><td>' .
$expSwap[3] . 'MB</td></tr>';

?>

</table>

<h1>SYSTEM PLIKÓW</h1>
<table class="zasoby">
    <tr><td>nazwa</td><td>wielkość</td><td>zajęte</td><td>podmontowany</td></tr>

<?php

$df = ereg_replace (' +', ' ', (shell_exec('df | head -n -1 | sed -e 1d')));
$expDf = explode(' ', $df);

echo '<tr><td>' . $expDf[0] . '</td><td>' . zaokr($expDf[3]*1024) . 'B</td><td>' .
$expDf[4] . '</td><td>' . $expDf[5] . '</td></tr>';

?>

</table>
```

Listing 3.6 zawartość pliku /var/www/zasoby.php

Skrypt *funkcje.php* zawiera dwie funkcje, używane w pozostałych skryptach

```

<?php

function logowanie() {
    session_start();
    $zalogowany = 0;

    if(isset($_POST['haslo'])) {
        $haslo = '527f6fbed55e9370dc394422b55814f8';
        if(md5($_POST['haslo']) == $haslo) {
            $_SESSION['zalogowany'] = 1;
        }
    }

    if(isset($_GET['wyloguj'])) {
        $_SESSION['zalogowany'] = 0;
    }

    if(isset($_SESSION['zalogowany']))
        $zalogowany = $_SESSION['zalogowany'];

    return $zalogowany;
}

function zaokr($liczba) {
    $ile = 0;
    while($liczba > 1024) {
        $liczba = $liczba / 1024;
        $ile++;
        if($ile == 4) break;
    }

    if($ile == 1)
        $jedn = 'K';
    elseif($ile == 2)
        $jedn = 'M';
    elseif($ile == 3)
        $jedn = 'G';
    elseif($ile == 4)
        $jedn = 'T';
    else
        $jedn = '';

    return round($liczba, 2) . $jedn;
}

```

Listing 3.7 zawartość pliku /var/www/funkcje.php

W pliku *style.css* umieszczone zostały reguły opisujące sposób prezentacji wszystkich powyższych stron www.

```

* { margin: 0px;
  font-size: 11px;
  font-family: Helvetica;
  color: #000000;}

body { text-align: center;
  background: url('tlo.gif');
  margin: auto; }

table { border-spacing: 0px;
  margin: 20px auto;
  border-collapse: collapse;
}

td { border: 1px #bbb solid;
  padding: 5px;
  text-align: center;}

a { text-decoration: none;
  font-weight: bold;}

h1 { font-size: 16px;
  margin-top: 20px;}

h2 { margin-bottom: 20px;}

p.pingok { color: green;
  font-weight: bold;}

p.pingbrak { color: red;
  font-weight: bold;}

.uslugi { display: inline;}

.zasoby { margin-top: 5px;
  font-size: 12px;}

#main { margin: 20px auto;
  background-color: #eee;
  width: 700px;
  border: 10px white solid;}

#naglowek { border-bottom: 10px white solid;
  text-align: left;
  padding: 2px;}

#czas { text-align: left;
  display: inline;
  margin-left: 0px;}

#menu { text-align: right;
  display: inline;
  word-spacing: 2px;
  margin-left: 532px;}

#logowanie { margin: 30px auto;}

#logowanie p { margin: 5px auto;}

#logowanie input { margin: 5px auto;}

#syslog { text-align: left;
  margin: 10px 40px 0px 40px;}

#ifconfig { text-align: left;
  margin: 10px 20px 10px 160px;}

```

Listing 3.8 zawartość pliku /var/www/style.css

3.2 Testy systemu

Zrealizowany serwer został poddany testom aby stwierdzić czy spełnia on założenia projektu. Przetestowano w nim bezpieczeństwo, wydajność oraz szybkość.

3.2.1 Nmap – skanowanie portów

Skanowanie portów to czynność polegająca na wysyłaniu pakietów TCP lub UDP do hosta. Skanowanie ma na celu określenie usług dostarczanych na danej maszynie oraz oprogramowania, które obsługuje dane usługi. Tak zdobyte informacje można wykorzystać do złamania zabezpieczeń systemu dlatego wszystkie porty z wyjątkiem tych na których działają usługi powinny być blokowane.

Na listingu 3.8 przedstawiono wynik skanowania portów serwera programem Nmap^[13] z jednego z hostów sieci lokalnej. Program został ustawiony na skanowanie metodą SYN wszystkich portów TCP dla adresu *192.168.1.245*. Metoda ta polega na wysyłaniu pakietów z flagą SYN a następnie oczekiwaniu na odpowiedź serwera. W zależności od otrzymanej flagi w pakiecie port jest uznawany za otwarty, zamknięty lub filtrowany. Dodatkowo zostały użyte opcje *-A* oraz *-v* wyświetlające informacje na temat systemu i usług. Aby skrócić wynik testu mniej znaczące linijki zastąpiono czerwonym wielokropkiem (...)

```
C:\> nmap -sS -A -v 192.168.1.254
```

```

Starting Nmap 5.00 ( http://nmap.org ) at 2009-08-28 17:43 Środkowoeuropejski czas letni
NSE: Loaded 30 scripts for scanning.
Initiating ARP Ping Scan at 17:43
Scanning 192.168.1.254 [1 port]
Completed ARP Ping Scan at 17:43, 0.30s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 17:43
Completed Parallel DNS resolution of 1 host. at 17:43, 0.00s elapsed
Initiating SYN Stealth Scan at 17:43
Scanning 192.168.1.254 [65535 ports]
Discovered open port 139/tcp on 192.168.1.254
Discovered open port 80/tcp on 192.168.1.254
Discovered open port 22/tcp on 192.168.1.254
Discovered open port 445/tcp on 192.168.1.254
Completed SYN Stealth Scan at 17:43, 0.39s elapsed (65535 total ports)
Initiating Service scan at 17:43
Scanning 4 services on 192.168.1.254
Completed Service scan at 17:43, 11.03s elapsed (4 services on 1 host)
Initiating OS detection (try #1) against 192.168.1.254
Retrying OS detection (try #2) against 192.168.1.254
Retrying OS detection (try #3) against 192.168.1.254
Retrying OS detection (try #4) against 192.168.1.254
NSE: Script scanning 192.168.1.254.
NSE: Starting runlevel 1 scan
Initiating NSE at 17:43
Completed NSE at 17:44, 63.23s elapsed
NSE: Starting runlevel 2 scan
Initiating NSE at 17:44
Completed NSE at 17:45, 32.10s elapsed
NSE: Script Scanning completed.
Host 192.168.1.254 is up (0.00018s latency).
Interesting ports on 192.168.1.254:
Not shown: 65531 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 5.1 (protocol 2.0)
80/tcp    open  http     lighttpd 1.4.22
|_ html-title: zarządzaj serwerem
139/tcp   open  netbios-ssn Samba smbd 3.X (workgroup: LOKALNA)
445/tcp   open  netbios-ssn Samba smbd 3.X (workgroup: LOKALNA)
MAC Address: 00:19:5B:7C:C4:F5 (D-Link)
No exact OS matches for host (If you know what OS is running on it, see
http://nmap.org/submit/ ).
TCP/IP fingerprint: ...

Network Distance: 1 hop

Host script results:
|_ nbstat: NetBIOS name: SERWER, NetBIOS user: <unknown>, NetBIOS MAC: <unknown>
|   ...
|_ smb-os-discovery: Unix
|   LAN Manager: Samba 3.3.7
|   Name: Unknown\Unknown
|_ System time: 2009-08-28 17:40:33 UTC+2
...

Nmap done: 1 IP address (1 host up) scanned in 131.41 seconds
Raw packets sent: 1281 (62.852KB) | Rcvd: 1036 (44.292KB)

```

Listing 3.9 rezultat skanowania portów serwera z sieci lokalnej

Program przeskanował w ciągu niewiele ponad dwóch minut 65535 portów na serwerze. Znalazł cztery otwarte porty:

- 22 – umożliwiający zdalne logowanie poprzez SSH
- 80 – pozwalający na zarządzanie serwerem przez www
- 139, 445 – służące do udostępniania plików za pomocą Samby

Drugie skanowanie przeprowadzono z komputera znajdującego się w Internecie. Ponownie przeskanowano wszystkie porty TCP według tej samej konfiguracji ale tym razem dla adresu publicznego *przeor.eu*

```
C:\> nmap -sS -A -v przeor.eu
```

```
Starting Nmap 5.00 ( http://nmap.org ) at 2009-08-28 19:46 Środkowoeuropejski czas letni
NSE: Loaded 30 scripts for scanning.
Initiating Ping Scan at 19:46
Scanning 89.74.169.240[8 ports]
Completed Ping Scan at 19:46, 0.27s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 19:46
Completed Parallel DNS resolution of 1 host. at 19:46, 0.02s elapsed
Initiating SYN Stealth Scan at 19:46
Scanning chello8974169240.chello.pl (89.74.169.240) [65535 ports]
Discovered open port 22/tcp on 89.74.169.240
Discovered open port 1723/tcp on 89.74.169.240
Discovered open port 3389/tcp on 89.74.169.240
Initiating Service scan at 20:15
Scanning 3 services on chello8974169240.chello.pl (89.74.169.240)
Completed Service scan at 20:15, 19.48s elapsed (3 services on 1 host)
Initiating OS detection (try #1) against chello8974169240.chello.pl (89.74.169.240)
Retrying OS detection (try #2) against chello8974169240.chello.pl (89.74.169.240)
Retrying OS detection (try #3) against chello8974169240.chello.pl (89.74.169.240)
Initiating Traceroute at 20:16
89.74.169.240: guessing hop distance at 4
Completed Traceroute at 20:16, 0.05s elapsed
Initiating Parallel DNS resolution of 6 hosts. at 20:16
Completed Parallel DNS resolution of 6 hosts. at 20:16, 0.02s elapsed
NSE: Script scanning 89.74.169.240.
NSE: Starting runlevel 1 scan
Initiating NSE at 20:16
NSE: Timing: About 50.00% done; ETC: 20:17 (0:00:31 remaining)
Completed NSE at 20:17, 81.38s elapsed
NSE: Script Scanning completed.
Host chello8974169240.chello.pl (89.74.169.240) is up (0.073s latency).
Interesting ports on chello8974169240.chello.pl (89.74.169.240):
Not shown: 65532 filtered ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 5.1 (protocol 2.0)
1723/tcp  open  pptp         linux (Firmware: 1)
3389/tcp  open  microsoft-rdp Microsoft Terminal Service
Device type: WAP|general purpose|broadband router|VoIP gateway
Running (JUST GUESSING) : Linux 2.4.X|2.6.X (91%), Motorola embedded (86%), Netgear
embedded (86%), Netcomm embedded (85%)
Aggressive OS guesses: DD-WRT v23 (Linux 2.4.36) (91%), Linux 2.6.18 (OSSIM) (91%),
Motorola SURFboard SB5100i cable modem (86%), Netgear WGR614v7 wireless broadband router
(86%), Netcomm V300 VoIP gateway (85%)
No exact OS matches for host (test conditions non-ideal).
Service Info: OS: Windows

...

Nmap done: 1 IP address (1 host up) scanned in 1869.78 seconds
Raw packets sent: 132314 (5.826MB) | Rcvd: 9674 (407.195KB)
```

Listing 3.10 rezultat skanowania portów serwera z Internetu

Podczas tego skanowania program znalazł tylko trzy otwarte porty:

- 22 – umożliwiający zdalne logowanie poprzez SSH
- 1723 – służący do nawiązywania połączeń PPTP przez Poptop
- 3389 – pozwalający na zdalnie podłączanie się do windowsowego serwera

Test ten pokazuje że na każdym z interfejsów sieciowych serwera otwarte są tylko niezbędne porty a ich ilość ograniczona jest do minimum dzięki czemu serwer jest mniej narażony na niebezpieczeństwa.

3.2.2 Ilość połączeń TCP oraz UDP

Główną wadą domowych routerów jest ich niestabilna praca podczas obsługi dużej ilości połączeń TCP i UDP. Po kilkunastogodzinnej pracy router przy korzystaniu z programów P2P najczęściej całkowicie się zawiesza: nie przekazuje pakietów i uniemożliwia nawet logowanie w celu jego zrestartowania. Dodatkowo w testowanym modelu D-link DI-524 przestał działać nawet przełącznik zapewniający wymianę danych między komputerami w sieci lokalnej. Najczęściej jedynym rozwiązaniem jest odłączenie routera od zasilania.

W celu przetestowania pod tym kątem realizowanego serwera na dwóch komputerach w sieci lokalnej włączono programy typu P2P pobierające i wysyłające duże ilości danych. Podczas testu za pomocą polecenia *netstat-nat* sprawdzano na serwerze ilość połączeń

```
$ netstat-nat -n | sed -e 1d | awk ' {print $4}' | sort | uniq -c
```

```
root@serwer:~ $ netstat-nat -n | sed -e 1d | awk ' {print $4}' | sort | uniq -c
  67 ASSURED
 120 ESTABLISHED
  23 TIME_WAIT
  28 UNREPLIED
```

Rys. 3.7. zrzut ekranu ilustrujący ilość aktywnych połączeń

W przypadku protokołu TCP średnia ilość ustanowionych połączeń wynosiła około 115, protokół UDP zapewniał średnio 70 połączeń. Po 48 godzinach test został przerwany ponieważ duże obciążenie łącz oraz znaczna ilość aktywnych połączeń nie miały negatywnego wpływu na funkcjonowanie serwera.

3.2.3 Czas uruchamiania systemu

Podczas awarii lub aktualizacji oprogramowania system wymaga czasami restartu. Należy zadbać o to aby był on jak najszybszy a użytkownicy sieci mieli jak najmniejszą przerwę w dostępie do usług. Tabela 4.1 przedstawia czas uruchamiania się kilku wybranych dystrybucji. Czas ten zmierzony został za pomocą programu Bootchart^[14]

Dystrybucja	Czas w sekundach	Komputer
-------------	------------------	----------

openSUSE 10.3 Beta 1	27	Intel Core Duo T2400 1GB RAM
Fedora 7	41	
PCLinuxOS 2007	32	
Kubuntu Tribe 4	31	
Mandriva 2008 Beta 1	29	
Core 2.0	21	Intel P4 2800MHz 512MB RAM

Tab. 3.1 czas uruchamiania się wybranych dystrybucji ²

Dzięki zastosowaniu możliwie małej ilości usług i opcji jądra system Core 2.0 uruchamia się dużo szybciej niż dostępne dystrybucje ze standardową konfiguracją. Na rysunku 3.8 przedstawiono wykres ilustrujący proces uruchamiania się zaprojektowanego systemu.

² dane z publikacji <http://francis.giannaros.org/blog/2007/08/23/comparing-distribution-boot-times/>



Rys. 3.8. wykres przedstawiający czas uruchamiania się procesów podczas startu systemu

3.2.4 Podsumowanie testów

Przeprowadzone testy nie wykazały luk w bezpieczeństwie systemu. Wydajność i szybkość serwera wypadła znacznie powyżej potrzeb sieci lokalnej składającej się z kilku komputerów. Zrealizowany serwer bez zarzutu pełnił swoje funkcje przez miesiąc czasu. Podczas tego okresu nie było żadnej potrzeby restartowania systemu ani interwencji administratora.

Podsumowanie i wnioski

Zrealizowany serwer znacznie usprawnia administrowanie siecią oraz dostarcza wielu funkcji jej użytkownikom. Dzięki zainstalowaniu minimalnej ilości programów system może pracować na starych komputerach, jest łatwy do aktualizacji i uporządkowany. Przygotowane skrypty oraz interfejs www umożliwiają wygodne zarządzanie serwerem. Wprowadzenie monitoringu sieci z podziałem na komputery ułatwia kontrolowanie ruchu generowanego przez poszczególnych użytkowników. Zainstalowany serwer Samby pozwala na szybką wymianę plików w sieci dzięki czemu komputery użytkowników chcących dzielić się plikami nie muszą być permanentnie włączone. Dostarczona usługa VPN może służyć nie tylko jako narzędzie umożliwiające łączenie się z siecią lokalną ale również jako mechanizm do szyfrowania danych gdy użytkownik korzysta z publicznego dostępu do Internetu.

Instalacja serwera w domu lub małej firmie wiąże się z niewielkim kosztem. Składa się na niego cena komputera, przełącznika oraz ewentualnie dodatkowej karty sieciowej. Warto zauważyć że zrealizowany serwer znacznie przekracza wydajnością oraz funkcjonalnością urządzenia dostępne na rynku w tej samej cenie.

Zaprojektowany serwer oparty został na systemie Linuks przez co jest wysoce skalowany, łatwo jest go uprościć bądź rozbudować o nowe funkcje. W przyszłości w systemie można byłoby zainstalować zarządzanego przez www klienta sieci BitTorrent, który umożliwiałby ściąganie plików z Internetu. Po ściągnięciu pliki udostępniane byłyby użytkownikom sieci lokalnej w jednym z zasobów Samby. Funkcją przydatną do zarządzania siecią osiedlową mógłby być mechanizm zmniejszający pasmo łącza danemu użytkownikowi po przekroczeniu limitu ściągniętych danych. W przypadku sieci firmowej system można byłoby rozbudować na przykład o serwer wydruku.

Bibliografia

- [1] Laurence Bonney: *Boot loader showdown: Getting to know LILO and GRUB* (<http://www.ibm.com/developerworks/linux/library/l-bootload.html>) 2005
- [2] Michael Rash: *Bezpieczeństwo sieci w Linuksie. Wykrywanie ataków i obrona przed nimi za pomocą iptables, psad i fw*, Wydawnictwo Helion 2008
- [3] Brian Komar: *Administracja sieci TCP/IP dla każdego*, Wydawnictwo Helion 2000
- [4] Radosław Sokół: *Slackware Linux*, Wydawnictwo Helion 2006
- [5] Richard Sharpe, Tim Potter, Jim Morris: *Samba dla każdego*, Wydawnictwo Helion 2002
- [6] Linux Magazine: *VPN dla każdego*, sierpień 2004, strona 52
- [7] Andi Gutmans, Stig Sæther Bakken and Derick Rethans: *PHP 5 Power Programming*, Prentice Hall 2004
- [8] Linux+: *Lighttpd serwer HTTP – alternatywa dla Apache*, listopad 2007, strona 44
- [9] Cezary M. Kruk: *Cron, czyli zadania na czas* (<http://webhosting.pl/Cron.czyli.zadania.na.czas>) 2007
- [10] Linux+: *Bezpieczna powłoka – OpenSSH*, styczeń 2008, strona 34
- [11] Free Software Foundation: *GNU General Public License* (<http://gnu.org.pl/text/licencja-gnu.html>) 2007
- [12] Douglas E. Comer: *Sieci komputerowe TCP/IP. Zasady, protokoły i architektura*, tom 1, WNT 1998
- [13] Gordon Fyodor Lyon: *Nmap Network Scanning - The Official Nmap Project Guide to Network Discovery and Security Scanning*, Nmap Project 2009
- [14] Ziga Mahkovec: *Bootchart documentation* (<http://www.bootchart.org/docs.html>) 2005