Politechnika Warszawska
Wydział Elektroniki i Technik Informacyjnych
Instytut Informatyki
Wieczorowe Studia Zawodowe - Informatyka

Rok akademicki: 2003/2004

Praca dyplomowa inżynierska

Bartosz Maciej Krajnik

Zapory sieciowe uwzględniające filtrowanie warstwy aplikacji i jakości usług

Opiekun pracy dyplomowej
dr inż. Jacek Wytrębowicz

Ocena: _____

_____
Podpis Przewodniczącego
Komisji Egzaminu Dyplomowego

Kierunek: Informatyka
Data urodzenia: 1977-10-26
Data rozpoczęcia studiów: 1997-02-01

Życiorys

Urodziłem się 26-go października 1977r. w Słupsku. Tam też ukończyłem Szkołę Podstawową. W latach 1992-1997 uczyłem się w Technikum Samochodowym w Koszalinie w klasie o specjalności Naprawa i Eksploatacja Pojazdów Samochodowych. Po zdaniu matury dostałem się na Wydział Elektroniki i Technik Informacyjnych Politechniki Warszawskiej na kierunek Elektronika. Po dwóch latach studiów zmieniłem specjalność na Informatykę.

Od kilku lat pracuję z sieciami LAN/WAN. Zajmuję się administrowaniem i bezpieczeństwem systemów UNIX/Linux/Solaris/BSD.

Spis moich prac i programów można znaleźć na stronie:

http://www.bmk-it.com/

_____
Podpis studenta

Egzamin dyplomowy

Złożył egzamin dyplomowy w dniu _____
z wynikiem _____
Ogólny wynik studiów _____
Dodatkowe wnioski i uwagi komisji _____

Streszczenie

Praca niniejsza opisuje działanie zapory sieciowej filtrującej w warstwie aplikacji i sterującej przepływem danych – QoS (*ang. Quality of Services*). Największa uwaga jest skupiona na rozwiązaniu typu Open Source oraz systemie operacyjnym GNU/Linux. W pracy omówiono również podstawowe protokoły komunikacji wraz z ich możliwościami.

Utworzyłem startowaną z płyty CD zaporę sieciową filtrującą pakiety w 7-ej warstwie modelu OSI oraz zapewniającej jakość usług poprzez mechanizm kontroli przepływu. Odpowiednie skrypty zapewniają autowykrywanie sprzętu i proste dla użytkownika konfigurowanie zapory sieciowej. Przy pierwszym starcie istnieje możliwość zapisania konfiguracji na dyskietce, po późniejszym mechanicznym zabezpieczeniu jej przed zapisem możemy startować system z już zapisanymi ustawieniami.

Słowa kluczowe: firewall, zapora sieciowa, model OSI, Linux, filtrowanie pakietów.

Firewalls with Filtering in Application Layer and Quality of Services

Abstract

This book describes firewalls with filtering in application layer and with Quality of Services. The main focus is given on Open Source software and GNU/Linux system. I included here description of basic protocols with their possibilities.

I have created a firewall which filters packets in application layer and which supports Quality of Services. The attached CD contains the firewall. At the first boot you can write a configuration on floppy disk, every next boot will read this configuration.

Keywords: firewall, OSI model, Linux, packet filtering.

Spis treści

Wstęp.

W dzisiejszym świecie technologie informatyczne są bezwarunkowo związane z naszą pracą. Praca ta niejednokrotnie wymaga dostępu do sieci internet wraz ze wszystkimi tego konsekwencjami.

Systemy kontroli zawarte w standardowych programach nie są w stanie zabezpieczyć je przed działaniami intruzów z zewnątrz. Podobnie brak jest takiego mechanizmu w sieciach wewnętrznych małych i dużych firm – sieć nie potrafi sama się bronić – wymagany jest mechanizm zapewniający to.

Celem niniejszej pracy jest stworzenie systemu firewall posiadającego zaawansowane mechanizmy kontroli przepływu zapewniające najwyższą jakość filtrowania danych i przy tym rozdzielające odpowiednio pasmo dostępu do sieci internet.

Praktyczna część obejmuje utworzenie bootowalnej płyty CD z odpowiednio spreparowanym systemem pozwalającym na wszystkie opisane powyżej czynności.

1. Podstawy działania zapór sieciowych.

Ten rozdział wprowadza do poznania celu działania zapór sieciowych oraz wyjaśnia istotę ich implementacji.

Co to jest zapora sieciowa?
Zapora sieciowa jest urządzeniem sieciowym położonym pomiędzy dwoma różnymi sieciami (zazwyczaj pomiędzy jakąś organizacją a internetem) i chroniącą pewne określone stacje znajdujące się w sieci wewnętrznej przed atakami z zewnątrz. Często mianem zapór sieciowych określa się pojedyncze stacje z zainstalowanym oprogramowaniem mającym na celu ochronę ich samych. Tematem tej pracy są jednak komputery z dwoma (lub więcej) kartami sieciowymi stojące na granicy dwóch (lub więcej) sieci i filtrujące dane pomiędzy nimi.

Co powinna robić zapora sieciowa?
Zapora sieciowa zapewnia, że cała komunikacja pomiędzy dwoma sieciami (wewnętrzną i zewnętrzną) będzie poddana polityce bezpieczeństwa firmy. Zapory sieciowe kierują i kontrolują komunikacją decydując, co przepuszczać a co odrzucać. Dodatkowo zabezpieczając zaufaną sieć przed internetem zapory sieciowe mogą odpowiednio chronić pewien wydzielony obszar sieci wewnętrznej lub indywidualne stacje.

Dlaczego firmy potrzebują zapór sieciowych?
Firmy na całym świecie wykorzystują technologie internetowe do prowadzenia operacji biznesowych. Zapory sieciowe pomagają firmom równoważyć otwarcie internetu z potrzebą zabezpieczenia prywatności i integralności komunikacji biznesowych.

Jak działają zapory sieciowe?
Historycznie są wykorzystywane trzy różne technologie implementacji zapór sieciowych: filtrowanie pakietów, bramy warstwy aplikacji i kontrola stanów.

Filtrowanie pakietów – zazwyczaj implementowane na ruterach, filtrują ruch w oparciu o zawartość pakietów, np. adresy IP. Sprawdzają pakiet w warstwie

sieciowej i są niezależne od aplikacji, zapewniają dobrą skalowalność i prędkość. Są najmniej zabezpieczającymi typami zapór sieciowych. Ponieważ nie są świadome aplikacji, nie mogą zrozumieć kontekstu danej komunikacji robiąc je łatwymi do złamania.

Bramy warstwy aplikacji – poprawiają bezpieczeństwo egzaminując całą warstwę aplikacji przenosząc kontekst informacji do procesu decydującego - jakkolwiek łamią zasadę modelu klient/serwer. Każda komunikacja klient/serwer wymaga tutaj dwóch połączeń: jednego od klienta do zapory sieciowej (która działa jako zastępstwo dla określonego serwera) i jednego od zapory sieciowej do rzeczywistego serwera. W dodatku każda aplikacja wymaga nowego „proxy" czyniąc skalowalność i obsługę problematyczną.

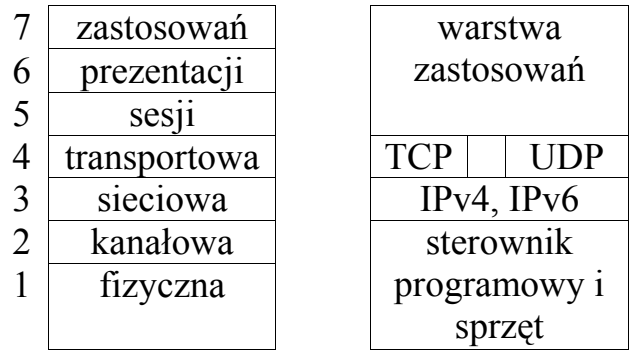
Kontrola stanów – wprowadza najwyższy poziom zabezpieczeń i przewyższa poprzednie dwie metody rozumiejąc filtrowanie warstwy aplikacji bez łamania modelu klient/serwer. Rozpoznaje stany połączeń ze wszystkich aplikacji i alokuje te informacje w dynamicznej tablicy. Wprowadza to rozwiązanie w pełni bezpiecznie i oferujące maksymalną prędkość, skalowalność oraz niezawodność.

2. Protokoły sieciowe i ich działanie

W tym rozdziale opisano zasadę działania protokołów oraz wyjaśniono budowę modelu sieci OSI. Wyjaśniono reguły na podstawie, których dane pakiety z danego protokołu powinny być przepuszczane i odrzucane przez zaporę sieciową.

2.1. Model sieci OSI

Sieć komputerowa została zdefiniowana na poszczególne warstwy. Owe warstwy zwykło się opisywać zgodnie z koncepcją modelu sieci OSI, czyli modelu Połączenia Systemów Otwartych (ang. *Open Systems Interconnection*), przyjętego jako norma ISO, tj. Międzynarodowej Organizacji Normalizacyjnej (ang. *International Organization for Standardization*). Model ten składa się z siedmiu warstw, które wraz z ich przybliżonym odwzorowaniem na rodzinę protokołów Internetu są przedstawione na rys 1.1.



Rys. 1.1. Warstwy modelu OSI oraz rodzina protokołów internetu.

Dwie dolne warstwy są obsługiwane przez sprzęt sieciowy oraz obsługujące ten sprzęt oprogramowanie (sterownik). Warstwa sieciowa jest obsługiwana przez oprogramowanie protokołów IPv4 oraz IPv6. Trzy górne warstwy modelu OSI są połączone w jedną warstwę zastosowań, czyli warstwę oprogramowania użytkowego. Do tej warstwy należą klienci WWW (przeglądarki), klienci Telnet, serwery WWW, serwey FTP, klienci SMTP lub jakiekolwiek inne programy użytkowe, z których korzystamy. Gdy używamy protokołów Internetu zazwyczaj

nie rozróżniamy trzech górnych warstw modelu OSI.

W pracy tej na szczególną uwagę zasługuje warstwa najwyższa, która zostanie dokładniej opisana poniżej wraz z ogólnym opisem pozostałych warstw.

Najniższa warstwa opisuje fizyczne połączenie komputerów. Następna to warstwa kanałowa (nazywana czasem warstwą łącza danych). Zawiera ona programy obsługi urządzeń wchodzące w skład systemu operacyjnego i odpowiadające im karty interfejsów sieciowych.

Warstwa trzecia to warstwa sieciowa. Obsługuje ona ruch pakietów w sieci. To w tej warstwie następuje rutowanie pakietów. W zestawie protokołów TCP/IP warstwę tę tworzą: IP (*Internet Protocol*), ICMP (*Internet Control Message Protocol*) i IGMP (*Internet Group Management Protocol*).

Warstwa czwarta (transportowa) zapewnia przepływ danych pomiędzy dwoma stacjami obsługując znajdującą się nad nią warstwę aplikacji. W skład TCP/IP wchodzą dwa całkowicie różne protokoły warstwy transportowej: TCP (*Transmission Control Protocol*) i UDP (*User Datagram Protocol*). W pisaniu programów sieciowych można całkowicie pominąć nagłówki TCP albo UDP i tworzyć gniazdo surowe (*raw socket*) – zainteresowanych odsyłam do pozycji [14] w spisie bibliografii.

Trzy ostatnie warstwy są połączone w jedną warstwę zastosowań, czyli warstwę oprogramowania użytkowego (używając internetu najczęściej nie rozróżnia się tych trzech warstw), obsługuje ona funkcje związane z określoną aplikacją korzystającą z sieci. To w tej warstwie rozróżnić można dane, jakie wysyłamy i odbieramy z internetu. To właśnie ta warstwa umożliwia nam najdokładniejsze filtrowanie, a co za tym idzie wymaga najwięcej pracy ze strony zapory.

2.2. TCP

Jednym z najważniejszych protokołów 4-ej warstwy modelu OSI mających niewątpliwie największe zastosowanie w dzisiejszym internecie jest TCP. Na nim bazuje HTTP (*Hypertext Transfer Protocol*) - usługi WWW, FTP (*File Transfer Protocol*) czy SMTP (*Simple Mail Transport Protocol*). TCP zapewnia zorientowaną połączeniowo, niezawodną obsługę ciągu przesyłanych bajtów. Określenie „zorientowany połączeniowo” oznacza, że dwie aplikacje używające TCP muszą, przed rozpoczęciem wymiany danych, nawiązać połączenie TCP. To dzięki TCP przemieszczanych jest ponad 80% danych. Wśród usług dostępnych tą metodą komunikacji na szczególną uwagę zasługuje FTP, SSH, TELNET, SMTP,

HTTP, POP3, IMAP.

Aby zrozumieć zasadę działania protokołu TCP, należy poznać sposób, w jaki jest nawiązywane połączenie TCP, jak odbywa się transport danych oraz kończenie sesji TCP. Bez tego nie będzie można zrozumieć, jakie dane powinniśmy akceptować a jakie odrzucać. Ustanawianie połączenia TCP przebiega według poniższego scenariusza:

- serwer musi być przygotowany na przyjęcie nadchodzącego połączenia. W tym celu wykonuje otwarcie bierne (ang.*passive open*),
- klient rozpoczyna otwarcie aktywne (ang. *active open*) połączenia. To powoduje, że oprogramowanie TCP klienta wysyła segment danych SYN (nazwa tego segmentu pochodzi od słowa ang. *synchronize*), zawierający początkowy numer kolejny danych, które ten klient będzie przesyłać przez to połączenie. Zazwyczaj w tym segmencie SYN nie przesyła się danych, zawiera on tylko nagłówek IP, nagłówek TCP i ewentualnie opcje TCP,
- serwer musi potwierdzić przyjęcie segmentu SYN od klienta i wysłać własny segment SYN, zawierający początkowy numer kolejny danych, które serwer będzie wysyłać przez to połączenie. Serwer wysyła w jednym segmencie SYN również potwierdzenie ACK (ang. *acknowledgment*).

Po nawiązaniu połączenia TCP obywa się niezawodny transport danych przy uwzględnieniu następujących zasad:

- dane przekazywane przez aplikację są dzielone na fragmenty, które według TCP mają najlepszy do przesłania rozmiar. Jednostka informacji przekazywana przez TCP do IP nazywana jest *segmentem*,
- kiedy TCP wysyła segment, to jednocześnie uruchamia zegar, który rozpoczyna oczekiwanie na potwierdzenie odebrania segmentu przez drugą stronę. Jeśli potwierdzenie nie nadejdzie w określonym czasie, segment wysyłany jest powtórnie,
- kiedy TCP odbiera dane nadesłane przez drugą stronę połączenia, wysyła wtedy potwierdzenie odbioru,
- TCP posługuje się sumami kontrolnymi nagłówka datagramu i danych. Jest to suma typu end-to-end, której zadaniem jest wykrycie każdej modyfikacji danych w czasie ich przesyłania. Jeśli segment zostanie nadesłany z niepoprawną sumą kontrolną, TCP odrzuca go i nie potwierdza odbioru (spodziewa się, że wysyłający prześle go po raz kolejny, po odczekaniu założonego czasu),
- ponieważ segmenty TCP są przesyłane jako datagramy IP, które mogą być

nadsyłane w dowolnej kolejności, to segmenty TCP również mogą nadchodzić w innej kolejności niż zostały wysłane. Strona odbierająca segmenty TCP porządkuje je, jeśli zachodzi taka konieczność, i przekazuje do aplikacji,
- ponieważ datagramy IP mogą być powielane, odbiorca TCP musi umieć odrzucić zdublowane dane,
- ponadto TCP zapewnia kontrolę przepływu. Każdy z końców połączenia TCP ma przestrzeń buforową o skończonej wielkości. Strona odbierająca TCP pozwala stronie nadającej na wysłanie tylko takiej ilości danych, która może być umieszczona w buforze odbiorcy. Taki mechanizm zapobiega przeładowaniu bufora w hoście wolnym przez datagramy ze stacji szybszej.

Podczas gdy do ustanowienia połączenia potrzeba trzech segmentów, to do zakończenia używa się czterech:
- najpierw program użytkowy wywołuje funkcję zamknięcia. Mówimy wówczas, że ten punkt końcowy połączenia wywołuje zamknięcie aktywne (ang. *active close*). Oprogramowanie TCP po tej stronie połączenia wysyła segment danych FIN (ang. *finish*), oznaczający zakończenie wysyłania danych,
- drugi punkt końcowy połączenia, który odbiera segment FIN, wykonuje zamknięcie bierne (ang. *passive close*). Oprogramowanie TCP potwierdza przyjęcie segmentu FIN. Informacja o otrzymaniu segmentu FIN jest również przesyłana do programu użytkowego jako znacznik końca pliku (po wszystkich danych, które mogą już oczekiwać w kolejce na pobranie przez program użytkowy), ponieważ odebranie segmentu FIN oznacza, że ten program użytkowy już nie otrzyma żadnych dodatkowych danych poprzez to połączenie,
- po pewnym czasie ten drugi program użytkowy, który odebrał znacznik końca pliku, wywołuje funkcję zamknięcia, aby zamknąć swoje gniazdo. To powoduje, że jego warstwa TCP wysyła segment FIN,
- oprogramowanie TCP w tym systemie, który odebrał ten ostatni segment FIN (w tym punkcie końcowym połączenia, który wykonuje zamknięcie aktywne), potwierdza przyjęcie segmentu FIN.

Na podstawie powyższego opisu możemy się zorientować, że najbezpieczniej jest przepuszczać segmenty nawiązujące połączenie TCP tylko z naszej sieci wewnętrznej, a wszystkie pozostałe odpowiednio przy założeniu, że odnoszą się one do już nawiązanej sesji TCP. Pojęcie stanów będzie omówione w punkcie 5.2.

W tym samym punkcie będzie omówiona konfiguracja firewall'a w oparciu o powyższy opis.

2.3. UDP

UDP jest prostym, wykorzystującym datagramy, protokołem warstwy transportowej: każda wysłana informacja powoduje utworzenie dokładnie jednego datagramu UDP, który z kolei powoduje wysłanie datagramu IP. Jest to podejście inne niż w przypadku zorientowanych strumieniowo protokołach, takich jak TCP, w których dane wysyłane przez aplikację mogą mieć niewiele wspólnego z tym, co faktycznie wysyłane jest w pojedynczym datagramie IP. Protokół UDP zapewnia większą prędkość transportu danych niż TCP, nie prowadzi on jednak kontroli dostarczanych pakietów i nie może być wykorzystywany podobnie jak TCP. Na nim bazują głównie programy interaktywne, aplikacje multimedialne czy system nazw DNS (*Domain Name System*).

Nie ma tutaj nic takiego jak nawiązywanie i kończenie połączenia (w protokole UDP nie istnieje takie pojęcie – jest on bezpołączeniowy). Na przykład klient UDP może utworzyć gniazdo i wysłać datagram do pewnego serwera, po czym może natychmiast przez to samo gniazdo wysłać drugi datagram do innego serwera. Podobnie serwer UDP może z jednego gniazda przyjąć pięć kolejnych datagramów, które nadeszły od pięciu kolejnych różnych klientów.

W UDP podobnie jak w protokole TCP najbezpieczniej jest przepuszczać datagramy nawiązujące połączenia z naszej sieci wewnętrznej oraz wszystkie inne, które są w relacji z już przepuszczonymi – odpowiednio stosuje się tutaj podejście stanów jak w protokole TCP.

2.4. ICMP

ICMP jest często rozpatrywany jako część warstwy IP. Protokół ten przesyła informacje o błędach i innych zajściach, które wymagają kontroli. Komunikaty ICMP są zwykle wysyłane przez warstwę IP lub protokoły warstw wyższych. Niektóre komunikaty ICMP zwracają błędy do procesów użytkownika.

2.5. Podsumowanie

W dzisiejszym internecie najwięcej danych jest transportowanych protokołem TCP i to na niego należy położyć największy nacisk przy tworzeniu zapór sieciowych. To właśnie przez TCP możliwe jest najwięcej nadużyć zarówno od strony klienta

jak i serwera. Właściwie skonfigurowana zapora sieciowa powinna zapewnić maksimum wydajności oraz bezpieczeństwa dla nieświadomych zagrożeń użytkowników internetu.

3. Funkcjonalne porównanie komercyjnych zapór sieciowych

W tym rozdziale dokonane zostanie porównanie zapór sieciowych w formie tabel. Zostaną one sklasyfikowane pod względem używanego oprogramowania, dostępnej architektury sprzętowej, zasady działania, zarządzalności oraz wydajności. Część tych danych pochodzi z udostępnionych zasobów firmy StoneSoft.

Poniższe dane stanowią krótki opis dostępnych na rynku firewall'i. Dzięki niemu możemy szybko zorinetować się, co do wyboru i jakości danego produktu. Jest to w pewnym stopniu uzasadnienie wybranej przeze mnie zapory sieciowej bazującej na systemie operacyjnym GNU/Linux.

3.1. CHECKPOINT NG

OGÓLNIE	
Obsługiwany OS	Solaris, Windows, Linux
Obsługiwany HW	Intel, SPARC
Maksymalna liczba interfejsów i ich typ	Zależny od OS
ZAPORA SIECIOWA	
Metoda inspekcji	Inspekcja zależna od stanów
Aplikacja zarządzająca protokołem	Nie
Aplikacja zarządzająca regułami	Nie
Podreguły (hierarchicznie)	Nie
Elementy zależne od kontekstu (aliasy)	Nie
Automatyczne generowanie reguł zapobiegających spoofingowi	Nie
VPN	
Zintegrowane z produktem	Tak
Algorytm szyfrujący	3DES, DES, AES, Cast
Multi-Link VPN	Nie
ZARZĄDZANIE	
System zarządzania	Tak
Zarządzanie bezpiecznym połączeniem	Szyfrowane
Logowanie	Własny system logowania
Ostrzeżenia	E-mail, SNMP, własna metoda
Raporty	3-cia strona SW
Wbudowana detekcja uszkodzeń	Limitowane
Platforma zarządzająca systemem	Solaris, Windows, Linux
Zdalny upgrade zapory sieciowej	Nie
WYDAJNOŚĆ DLA POJEDYŃCZEGO INTERFEJSU	
Przepustowość zapory sieciowej	765 Mbps
Przepustowość VPN (3DES/AES)	65 Mbps / 98 Mbps
Obsługiwanych połączeń w jednym czasie	1,000,000
WYSOKA DOSTĘPNOŚĆ / LOAD BALANCING	
Wbudowane w zaporę sieciową HA/LB	Tak/Nie
Wbudowany load balancing do ISP	Nie
Wbudowane w serwer „pool load sharing”	Nie

3.2. Cisco PIX 535

OGÓLNIE	
Obsługiwany OS	Cisco IOS
Obsługiwany HW	Zastosowany
Maksymalna liczba interfejsów i ich typ	10 Fast Ethernet albo 8 interfejsów Gigabit
ZAPORA SIECIOWA	
Metoda inspekcji	Inspekcja zależna od stanów
Aplikacja zarządzająca protokołem	Nie
Aplikacja zarządzająca regułami	Nie
Podreguły (hierarchicznie)	Nie
Elementy zależne od kontekstu (aliasy)	Nie
Automatyczne generowanie reguł zapobiegających spoofingowi	Nie
VPN	
Zintegrowane z produktem	Tak
Algorytm szyfrujący	3DES, DES
Multi-Link VPN	Nie
ZARZĄDZANIE	
System zarządzania	Tak
Zarządzanie bezpiecznym połączeniem	Tak
Logowanie	Logowanie systemowe
Ostrzeżenia	SNMP
Raporty	Wymaga menedżera urządzenia PIX
Wbudowana detekcja uszkodzeń	Nie
Platforma zarządzająca systemem	Bazująca na Web GUI
Zdalny upgrade zapory sieciowej	Nie
WYDAJNOŚĆ DLA POJEDYŃCZEGO INTERFEJSU	
Przepustowość zapory sieciowej	1,0 Gbps
Przepustowość VPN (3DES/AES)	100 Mbps / N/A
Obsługiwanych połączeń w jednym czasie	500,000
WYSOKA DOSTĘPNOŚĆ / LOAD BALANCING	
Wbudowane w zaporę sieciową HA/LB	Tak/Nie
Wbudowany load balancing do ISP	Nie
Wbudowane w serwer „pool load sharing”	Nie

3.3. Netscreen-1000

OGÓLNIE	
Obsługiwany OS	Zintegrowany OS
Obsługiwany HW	Zastosowany
Maksymalna liczba interfejsów i ich typ	2 Gigabit Ethernet, Zarządzany port (10/100Base-T Ethernet), Odseparowany port wysokiego dostępu
ZAPORA SIECIOWA	
Metoda inspekcji	Inspekcja zależna od stanów
Aplikacja zarządzająca protokołem	Nie
Aplikacja zarządzająca regułami	Nie
Podreguły (hierarchicznie)	Nie
Elementy zależne od kontekstu (aliasy)	Nie
Automatyczne generowanie reguł zapobiegających spoofingowi	Nie
VPN	
Zintegrowane z produktem	Tak
Algorytm szyfrujący	3DES, DES, AES
Multi-Link VPN	Nie
ZARZĄDZANIE	
System zarządzania	Tak
Zarządzanie bezpiecznym połączeniem	Opcjonalne
Logowanie	Logowanie lokalnego systemu
Ostrzeżenia	E-mail, pager, SNMP, własna metoda
Raporty	3-cia strona SW
Wbudowana detekcja uszkodzeń	Limitowane
Platforma zarządzająca systemem	Solaris, Windows
Zdalny upgrade zapory sieciowej	Nie
WYDAJNOŚĆ DLA POJEDYŃCZEGO INTERFEJSU	
Przepustowość zapory sieciowej	2,0 Gbps
Przepustowość VPN (3DES/AES)	1,0 Gbps / 1,0 Gbps
Obsługiwanych połączeń w jednym czasie	500,000
WYSOKA DOSTĘPNOŚĆ / LOAD BALANCING	
Wbudowane w zaporę sieciową HA/LB	Tak/Nie
Wbudowany load balancing do ISP	Nie
Wbudowane w serwer „pool load sharing”	Nie

3.4. SonicWALL GX650

OGÓLNIE	
Obsługiwany OS	Zintegrowany OS
Obsługiwany HW	Zastosowany
Maksymalna liczba interfejsów i ich typ	3 FastEthernet, 3 Gigabit (dostępne sloty rozszerzeń)
ZAPORA SIECIOWA	
Metoda inspekcji	Inspekcja zależna od stanów
Aplikacja zarządzająca protokołem	Nie
Aplikacja zarządzająca regułami	Nie
Podreguły (hierarchicznie)	Nie
Elementy zależne od kontekstu (aliasy)	Nie
Automatyczne generowanie reguł zapobiegających spoofingowi	Nie
VPN	
Zintegrowane z produktem	Tak
Algorytm szyfrujący	3DES, DES, ARCFour
Multi-Link VPN	Nie
ZARZĄDZANIE	
System zarządzania	Tak
Zarządzanie bezpiecznym połączeniem	Tak
Logowanie	Logowanie systemowe
Ostrzeżenia	E-mail, pager
Raporty	3-cia strona SW
Wbudowana detekcja uszkodzeń	Nie
Platforma zarządzająca systemem	Bazujący na Web GUI
Zdalny upgrade zapory sieciowej	Nie
WYDAJNOŚĆ DLA POJEDYŃCZEGO INTERFEJSU	
Przepustowość zapory sieciowej	1,67 Gbps
Przepustowość VPN (3DES/AES)	285 Mbps / N/A
Obsługiwanych połączeń w jednym czasie	500,000
WYSOKA DOSTĘPNOŚĆ / LOAD BALANCING	
Wbudowane w zaporę sieciową HA/LB	Tak/Nie
Wbudowany load balancing do ISP	Nie
Wbudowane w serwer „pool load sharing”	Nie

3.5. StoneGate

OGÓLNIE	
Obsługiwany OS	Zintegrowany OS
Obsługiwany HW	Intel, SPARC, Zastosowany
Maksymalna liczba interfejsów i ich typ	256 w każdej kombinacji z Ethernet, FastEthernet i GigabitEthernet
ZAPORA SIECIOWA	
Metoda inspekcji	Multi-layer Inspekcja zależna od stanów
Aplikacja zarządzająca protokołem	Tak
Aplikacja zarządzająca regułami	Tak
Podreguły (hierarchicznie)	Tak
Elementy zależne od kontekstu (aliasy)	Tak
Automatyczne generowanie reguł zapobiegających spoofingowi	Tak
VPN	
Zintegrowane z produktem	Tak
Algorytm szyfrujący	3DES, DES, AES, Cast, Blowfish
Multi-Link VPN	Tak
ZARZĄDZANIE	
System zarządzania	Tak
Zarządzanie bezpiecznym połączeniem	Tak
Logowanie	Relacyjna baza danych i archiwizacja bezpośrednia
Ostrzeżenia	E-mail, pager, SNMP, SMS, własna metoda
Raporty	3rd party SW (.csv & .xml formats)
Wbudowana detekcja uszkodzeń	Tak
Platforma zarządzająca systemem	Windows, Solaris, Linux
Zdalny upgrade zapory sieciowej	Tak
WYDAJNOŚĆ DLA POJEDYŃCZEGO INTERFEJSU	
Przepustowość zapory sieciowej	1,45 Gbps
Przepustowość VPN (3DES/AES)	80 Mbps / 180 Mbps
Obsługiwanych połączeń w jednym czasie	1,200,000
WYSOKA DOSTĘPNOŚĆ / LOAD BALANCING	
Wbudowane w zaporę sieciową HA/LB	Tak/Tak
Wbudowany load balancing do ISP	Tak
Wbudowane w serwer „pool load sharing”	Tak

3.6. Symantec Enterprise Firewall 7.0

OGÓLNIE	
Obsługiwany OS	Solaris, Windows
Obsługiwany HW	Intel, SPARC
Maksymalna liczba interfejsów i ich typ	Zależny od OS
ZAPORA SIECIOWA	
Metoda inspekcji	Hybrydowy, filtrowanie zależne od stanów i proxy
Aplikacja zarządzająca protokołem	Serwisy proxy
Aplikacja zarządzająca regułami	Nie
Podreguły (hierarchicznie)	Nie
Elementy zależne od kontekstu (aliasy)	Nie
Automatyczne generowanie reguł zapobiegających spoofingowi	Nie
VPN	
Zintegrowane z produktem	Tak
Algorytm szyfrujący	3DES, DES, AES
Multi-Link VPN	Nie
ZARZĄDZANIE	
System zarządzania	Tak
Zarządzanie bezpiecznym połączeniem	CLI:Tak, GUI:Nie
Logowanie	Własny system logowania
Ostrzeżenia	SNMP, E-mail, Pager, własna metoda
Raporty	3-cia strona SW
Wbudowana detekcja uszkodzeń	Nie
Platforma zarządzająca systemem	Solaris, Windows
Zdalny upgrade zapory sieciowej	Nie
WYDAJNOŚĆ DLA POJEDYŃCZEGO INTERFEJSU	
Przepustowość zapory sieciowej	100 Mbps
Przepustowość VPN (3DES/AES)	70 Mbps / N/A
Obsługiwanych połączeń w jednym czasie	N/A
WYSOKA DOSTĘPNOŚĆ / LOAD BALANCING	
Wbudowane w zaporę sieciową HA/LB	Nie/Nie
Wbudowany load balancing do ISP	Nie
Wbudowane w serwer „pool load sharing”	Nie

3.7. GNU/Linux

OGÓLNIE	
Obsługiwany OS	GNU/Linux
Obsługiwany HW	Intel, SPARC
Maksymalna liczba interfejsów i ich typ	Bez ograniczeń
ZAPORA SIECIOWA	
Metoda inspekcji	Hybrydowy, filtrowanie zależne od stanów i proxy
Aplikacja zarządzająca protokołem	Tak
Aplikacja zarządzająca regułami	Tak
Podreguły (hierarchicznie)	Tak
Elementy zależne od kontekstu (aliasy)	Tak
Automatyczne generowanie reguł zapobiegających spoofingowi	Tak
VPN	
Zintegrowane z produktem	Tak
Algorytm szyfrujący	3DES, DES, AES, Blowfish, SHA
Multi-Link VPN	Tak
ZARZĄDZANIE	
System zarządzania	Tak
Zarządzanie bezpiecznym połączeniem	CLI:Tak, GUI:Tak
Logowanie	Syslog
Ostrzeżenia	SNMP, E-mail, Pager, własna metoda
Raporty	3-cia strona SW
Wbudowana detekcja uszkodzeń	Tak
Platforma zarządzająca systemem	Linux
Zdalny upgrade zapory sieciowej	Tak
WYDAJNOŚĆ DLA POJEDYŃCZEGO INTERFEJSU	
Przepustowość zapory sieciowej	Zależna sprzętu
Przepustowość VPN (3DES/AES)	Zależna sprzętu
Obsługiwanych połączeń w jednym czasie	Bez ograniczeń
WYSOKA DOSTĘPNOŚĆ / LOAD BALANCING	
Wbudowane w zaporę sieciową HA/LB	Tak/Tak
Wbudowany load balancing do ISP	Tak
Wbudowane w serwer „pool load sharing”	Tak

3.8. Podsumowanie

Wśród komercyjnych rozwiązań na szczególną uwagę zasługuje StoneGate głównie z racji ilości dostępnych opcji. Ważny podkreślenia jest fakt możliwości zdalnego upgrade'u zapory sieciowej. Przyśpiesza to proces aktualizacji tym samym minimalizując prawdopodobieństwo złamania zabezpieczeń.

W opisie niniejszym nie zawarłem cen dostępnych rozwiązań, co jest na pewno kluczową sprawą przy podejmowaniu jakichkolwiek decyzji.

Wiele ludzi zdaje sobie sprawę z dostępności łatwego interfejsu konfigurującego zaporę, jednak osoba czyniąca to nie zawsze posiada odpowiednie kwalifikacje, co zazwyczaj kończy się dostarczeniem nieodpowiedniego produktu do klienta. Sytuacja taka nie może mieć miejsca przy oprogramowaniu bazującym na systemie Linux i produktach Open Source. Konfiguracja wymaga dogłębnej znajomości działania protokołów i możliwości nadużyć z ich strony. Poprawnie skonfigurowana zapora sieciowa nie stwarza możliwości złamania jej przy pojawieniu się nowych usług lub protokołów.

Wartą podkreślenia zaporą sieciową jest również „Checkpoint NG", który zawiera zintegrowane wsparcie dla wielu metod uwierzytelniania tożsamości użytkowników. Może to być realizowane za pomocą inteligentnych kart Smart Card, rozwiązań opartych na tokenach jak np. ActivCard, czy SecurID, haseł zapisywanych w bazach LDAP, serwerów RADIUS lub TACACS+, współdzielonych informacjach poufnych, certyfikatów cyfrowych X.509, a nawet różnorodnych technik biometrycznych. Dodatkowo Checkpoint obsługuje Secure Authentication API (SAA) - otwarty interfejs programowania aplikacji, który pozwala innym producentom na sprawne zintegrowanie własnych rozwiązań. Niewątpliwie dużą zaletą jest dla Checkpointa Obsługa Infrastruktury Klucza Publicznego. Zapewnia ona odpowiednie środowisko zarządzania dla dużych zastosowań sieci VPN (IPSec/IKE), pozwalając na bezpieczne i wydajne wykorzystanie i zarządzanie kluczami kryptograficznymi oraz certyfikatami cyfrowymi. Stosując się do standardów przemysłowych takich jak X.509, PKCS #11 i PKCS #12, „Checkpoint NG" daje duże możliwości instytucjom rozszerzającym swoje sieci intranetowe, ponadto Checkpoint umożliwia tworzenie hetero-genicznych sieci Extranet przez jednoczesną obsługę certyfikatów cyfrowych pochodzących od różnych Urzędów Certyfikacji (Certificate Authority, CA). Możliwość ta jest absolutnie krytyczna dla efektywnego wykorzystania sieci VPN integrujących wiele firm, gdyż każda firma może mieć różne rozwiązanie VPN i PKI. Współbieżne wsparcie certyfikatów pozwala na to, by pojedynczy

system „Checkpoint NG" jednocześnie ustanawiał wielokrotne połączenia IPSec z systemami wykorzystującymi rozwiązania VPN i PKI od różnych dostawców. System zabezpieczeń Checkpoint NG oprócz możliwości obsługi zewnętrznych Urzędów Certyfikacji posiada własny wbudowany moduł Internal Certificate Authority (ICA). Moduł ICA dostarcza certyfikaty cyfrowe X.509 dla zapewniania bezpieczeństwa wewnętrznej komunikacji sieciowych modułów VPN-1 oraz uwierzytelniania komunikacji IPSec/IKE. Do tego należy dodać bardzo elastyczny i przejrzysty interfejs użytkownika, czego brakuje innym rozwiązaniom.

Niestety „Checkpoint NG" nie posiada możliwości load-balancing'u, który w ostatnich czasach jest dosyć często wykorzystywany zarówno w małych jak i dużych firmach. To właśnie load-balancing zapewnia niezawodność oraz trwałość połączeń internetowych i dostęp do internetu. Wg mnie brak tej ostatniej cechy znacznie obniża wartość zapory sieciowej, nawet biorąc pod uwagę intuicyjny interfejs konfiguracyjny.

Linux niestety nie zapewnia takiej różnorodnności uwierzytelniania. Wiele z wyżej wymienionych jest tutaj zapewniana poprzez niezależne oprogramowanie. Wiele z tych programów jest dostępnych na zasadzie komercyjnej. Część opcji które są obsługiwane przez w.w. zapory sieciowe jest obecnie w trakcie powstawania na system operacyjny Linux. Należy mieć cały czas na uwadze fakt, że ten Open Source'owy system rozpoczął swoje istnienie zaledwie kilka lat temu i niektóre projekty będące jeszcze w fazie testowej mogą znacząco pokazać przewagę tego rozwiązania nad innymi.

4. Open Source a komercyjna zapora sieciowa Cisco - - porównanie funkcjonalne.

Ten rozdział opisuje eksperyment porównujący dwie zapory sieciowe ze zdolnością zestawiania prywatnych sieci wirtualnych (VPN) poprzez internet. Jedna jest rozwiązaniem Open Source, druga zaś produktem komercyjnym. Rozwiązania typu Open Source są tańsze i oferują większą kontrolę, natomiast rozwiązania komercyjne są droższe i reklamowane jako lepsze i bardziej elastyczne. Rozwiązania Open Source podążają za zasadą, że im więcej użytkowników będzie pracowało z kodem, tym bardziej będzie on bezpieczny i tym ciężej będzie go skompromitować. Komercyjne rozwiązania argumentują ukrywanie kodu źródłowego jako zmniejszenie prawdopodobieństwa na złamanie systemu. Dane dotyczące zapory sieciowej Cisco pochodzą z „Illinois State University - Department of Applied Computer Science".

4.1. Porównanie wybranych zapór sieciowych

Pierwsza zapora sieciowa użyta w tym porównaniu jest skonstruowana używając rozwiązania Open Source dostępnego dla systemu Debian Woody 3.0 GNU/Linux i na procesorze Pentium z dwoma kartami sieciowymi. Druga zapora sieciowa była zbudowana używając routera Cisco 2621 z systemem Cisco IOS. Całe oprogramowanie użyte na systemie Linux jest wolno dostępne na zasadzie Open Source i może być używane na różnych platformach sprzętowych, podczas gdy Cisco 2621 jest produktem komercyjnym obejmującym zarówno sprzęt jak i oprogramowanie.

4.2. Filtrowanie danych sieciowych w systemie Linux

System Linux z jądrem 2.4 ma zaimplementowany filtr pakietów netfilter - nazywany iptables. Użyto wersji iptables-1.2.6a w połączeniu z systemem Debian GNU/Linux w wersji Woody 3.0. Iptables dokonywało rewizji wg następujących kryteriów: adres IP źródłowy/przeznaczenia, protokół, nazwa DNS, interfejs i stan połączenia.

4.3. Filtrowanie danych sieciowych w systemach Cisco

Cisco IOS ma wbudowaną obsługę filtrowania trzech typów: statyczne filtrowanie pakietów (może być użyte przy użyciu standardowej/rozszerzonej listy dostępu do filtrowania innych protokołów), zwrotnej listy dostępu (jest używane do prowadzenia filtrowania w oparciu o stany pakietów przy użyciu Content Based Access Control - CBAC). Zwrotne listy dostępu są używane, aby zezwolić użytkownikom w zabezpieczonej sieci elastycznie wysyłać dane i odpowiednio przepuszczać pakiety przychodzące w odpowiedzi. System Cisco IOS ma właściwość znaną jako dostęp „Lock and Key", która łączy uwierzytelnianie hasłem z filtrowaniem pakietów opartym na stanach połączeń. Użytkownik musi wykonać poprawny proces uwierzytelniania do routera albo innego serwera (Radius albo TACACS+) zanim zapora sieciowa zezwoli na przepuszczanie danych.

4.4. Porównanie filtrowania warstwy aplikacji

Celem obsłużenia filtrowania w warstwie aplikacji zapora sieciowa zbudowana na systemie operacyjnym Linux używa oprogramowania „TIS firewall toolkit" w wersji 2.1. „TIS firewall toolkit" jest zbiorem aplikacji pośredniczących, które prowadzą dodatkową kontrolę wraz z logowaniem takich protokołów jak WWW, FTP, TELNET i SMTP. Jest również dostępna aplikacja „HTTP-Gopher proxy" i może być ona skonfigurowana do obsługi filtrowania stron WWW opartych o takie technologie jak Java, JavaScript czy ActiveX w zależności od stacji i od poziomu zabezpieczeń jego przeglądarki internetowej. Aplikacja „TIS firewall toolkit" poprzez pośredniczenie w protokole SMTP poprawia bezpieczeństwo wewnętrznego serwera SMTP czyniąc go trudnym do złamania podczas ataku. Zapora sieciowa Cisco IOS scala kilka dodatków poziomu aplikacji zwiększając bezpieczeństwo połączeń. Pierwszym z nich jest blokowanie Java w zastosowaniu do bloków apletów Javy dla ruchu webowego przy użyciu listy dostępu. Ta implementacja blokowania Javy ma jedną wadę w stosunku do Linuxa: aplikacja „TIS firewall toolkit" ma zdolność do współpracy z przeglądarką użytkownika bez potrzeby tworzenia oddzielnej listy zaufanych stron. Zapora sieciowa Cisco IOS nie ma zdolności do filtrowania aplikacji ActiveX.

4.5. Porównanie zdolności VPN

Dla zapory sieciowej Linux w celu implementacji VPN wybrano aplikację VPND wraz z SSH przy użyciu protokołu PPP ze względu na mocny algorytm szyfrujący

i scalony schemat kompresji. Aplikacja VPND używa szyfrowania typu Blowfish ze 128 bitowym algorytmem. Jeśli wymagana jest obsługa w oparciu o IPSEC, dostępna jest aplikacja Free S/WAN. Oprogramowanie Cisco IOS jest zdolne do utworzenia VPN-ów używając technologii szyfrującej Cisco (CET) lub IPSEC. Standardowe IPSEC jest skalowalne, co czyni je bardziej korzystne dla firm z sieciami heterogenicznymi. Technologia CET jest własnością firmy Cisco, która bazuje na standardowej sygnaturze cyfrowej (DSS), algorytmie klucza publicznego Diffie-Hellman'a (DH) i standardzie kodowania danych (DES).

4.6. Rezultaty eksperymentów

Dokonałem serii pomiarów bezpośrednio porównując zaporę sieciową bazującą na systemie operacyjnym Linux z zaporą sieciową Cisco IOS. Zmieniałem przy tym liczbę wpisów w każdej zaporze i rozmiar pakietów. Używałem transakcji TCP/UDP oraz strumieni TCP/UDP. Transakcje TCP są najbardziej miarodajną oceną wydajności zapory sieciowej biorąc pod uwagę źródło danych przepływającego przez zaporę sieciową pakietu o określonym rozmiarze (pamiętając o liczbie wpisów) oraz pakiet powracający do źródła poprzez tą samą zaporę sieciową z tą samą liczbą wpisów. Wpisy do zapory sieciowej były przewidywane dla najgorszych przypadków tak, aby wszystkie reguły były sprawdzone w każdym przypadku. Standardowa zapora sieciowa Cisco 2621 posiada 50 MHz procesor RISC z 50 M pamięci. Zapora sieciowa LINUX posiada 350 MHz procesor Pentium PC z 96 M pamięci.

Rozmiar pakietu = 1 bajt					
Ilość wpisów	**0**	**50**	**100**	**150**	**200**
Cisco	2541	1604	1434	1312	1216
Linux	5277	4293	4377	4033	3574
Rozmiar pakietu = 128 bajtów					
Ilość wpisów	**0**	**50**	**100**	**150**	**200**
Cisco	571	120	119	118	116
Linux	724	718	705	694	670

Tabela 3.1. Średnia liczba transakcji TCP (na sekundę).

Tabela 3.1. pokazuje, że zapora sieciowa Linux ma stosunkowo wyższą przepustowość transakcji dla ilości wpisów pomiędzy 0 a 200 i rozmiarów pakietów: 1 bajt i 128 bajtów. Były robione wielokrotne testy w celu osiągnięcia

99% pewności, co do różnicy pomiędzy tymi wartościami.

4.7. Podsumowanie

Ten eksperyment pokazuje, że dla tego specyficznego przykładu zapora sieciowa Linux ma znaczącą przewagę w obsługiwaniu transakcji, posiada także większe możliwości filtrowania w warstwie aplikacji. Zapora sieciowa Cisco IOS jest bardziej funkcjonalna przy filtrowaniu w warstwie sieciowej, jest zintegrowana z heterogenicznym wielo-protokołowym środowiskiem i skalowalna do obsługi PKI. Nie powinniśmy pomijać, że kilka projektów pod środowisko Linux znajdujących się jeszcze w fazie eksperymentalnej może dużo zmienić. Ostatecznie najbardziej efektywnym rozwiązaniem może być kombinacja obu – filtrowania w warstwie aplikacji i w warstwie sieciowej – i takie właśnie rozwiązanie jest tematem tej pracy.

5. Konfiguracja zapory sieciowej bazującej na systemie operacyjnym Linux

W tym rozdziale opisano konfigurację zapory sieciowej bazującej na systemie operacyjnym GNU/Linux Debian. Włączyłem do tego rozdziału opis konfiguracji wszystkich usług działających w systemie, konfigurację samego systemu, jego jądra oraz skrypty, które dodatkowo zabezpieczają system przed jego nadużyciem w przypadku czyjegoś udanego ataku.

5.1. Budowa systemu startującego z płyty CD

Do budowy firewall'a posłużyłem się systemem Linux GNU/Debian. Wybrałem odpowiednią partycję do budowy systemu (maksymalnie 650 MB pojemności) i zainstalowałem na niej standardową dystrybucję systemu, której użyłem później do budowy boot'owalnego z płytki firewall'a.

Po standardowej instalacji systemu zabezpieczyłem go. Tak wyglądało wyjście polecenia `netstat` i `ps` przed zabezpieczeniem:

```
# netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:2048            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:515             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:37              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:9               0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:13              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:113             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:25              0.0.0.0:*               LISTEN
udp        0      0 0.0.0.0:2048            0.0.0.0:*
udp        0      0 0.0.0.0:9               0.0.0.0:*
udp        0      0 0.0.0.0:111             0.0.0.0:*
udp        0      0 0.0.0.0:755             0.0.0.0:*
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
unix  4      [ ]         DGRAM                    143    /dev/log
unix  2      [ ACC ]     STREAM     LISTENING     292    /dev/printer
unix  2      [ ]         DGRAM                    285
unix  2      [ ]         DGRAM                    174

# ps axf
  PID TTY      STAT   TIME COMMAND
```

```
    6 ?        SW     0:00 [kupdated]
    5 ?        SW     0:00 [bdflush]
    4 ?        SW     0:00 [kswapd]
    3 ?        SWN    0:00 [ksoftirqd_CPU0]
    1 ?        S      0:03 init [2]
    2 ?        SW     0:00 [keventd]
    7 ?        SW     0:00 [i2oevtd]
   92 ?        S      0:00 /sbin/portmap
  147 ?        S      0:00 /sbin/syslogd
  150 ?        S      0:00 /sbin/klogd
  155 ?        S      0:00 /sbin/rpc.statd
  163 ?        S      0:00 /usr/sbin/inetd
  167 ?        S      0:00 /usr/sbin/lpd
  177 ?        S      0:00 /usr/sbin/atd
  180 ?        S      0:00 /usr/sbin/cron
  183 tty1     S      0:00 -bash
  184 tty2     S      0:00 /sbin/getty 38400 tty2
  185 tty3     S      0:00 /sbin/getty 38400 tty3
  186 tty4     S      0:00 /sbin/getty 38400 tty4
  187 tty5     S      0:00 /sbin/getty 38400 tty5
  188 tty6     S      0:00 /sbin/getty 38400 tty6
  206 ?        SW     0:00 [eth0]
  208 ?        SW     0:00 [eth1]
```

Pierwszą czynnością było wyłączenie niepotrzebnie działających demonów. Wykonałem następujące czynności:

```
# find /etc/rc* -name "*portmap" -exec rm {} \;
# find /etc/rc* -name "*nfs-kernel-server" -exec rm {} \;
# find /etc/rc* -name "*nfs-common" -exec rm {} \;
# find /etc/rc* -name "*atd" -exec rm {} \;
# find /etc/rc* -name "*lpd" - exec rm {} \;
# find /etc/rc* -name "*sysklog" -exec rm {} \;
# find /etc/rc* -name "*inetd" -exec rm {} \;
# find /etc/rc* -name "*cron" -exec rm {} \;
```

Po zrestartowaniu systemu uzyskałem:

```
# netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
unix  2      [ ]         DGRAM                    174
# ps axf
  PID TTY      STAT   TIME COMMAND
    6 ?        SW     0:00 [kupdated]
    5 ?        SW     0:00 [bdflush]
    4 ?        SW     0:00 [kswapd]
    3 ?        SWN    0:00 [ksoftirqd_CPU0]
```

```
    1 ?        S      0:03 init [2]
    2 ?        SW     0:00 [keventd]
    7 ?        SW     0:00 [i2oevtd]
  150 ?        S      0:00 /sbin/klogd
  183 tty1     S      0:00 -bash
  184 tty2     S      0:00 /sbin/getty 38400 tty2
  185 tty3     S      0:00 /sbin/getty 38400 tty3
  186 tty4     S      0:00 /sbin/getty 38400 tty4
  187 tty5     S      0:00 /sbin/getty 38400 tty5
  188 tty6     S      0:00 /sbin/getty 38400 tty6
  206 ?        SW     0:00 [eth0]
  208 ?        SW     0:00 [eth1]
```

Zobaczyłem co nasłuchuje na gnieździe w domenie unixowej:

```
# lsof -n|grep 174
klogd     150 root    1u  unix 0xc0fd8660              174 socket
```

Tak zabezpieczony system jest w zasadzie nie do zdobycia – nie ma możliwości dostępu z zewnątrz.
Następnie zainstalowałem uaktualnienia:

```
# apt-get update
Reading Package Lists... Done
Building Dependency Tree... Done
# apt-get upgrade
Reading Package Lists... Done
Building Dependency Tree... Done
0 packages upgraded, 0 newly installed, 0 to remove and 0  not upgraded.
```

Ściągnąłem najświeższe stabilne jądro (wtedy 2.4.23) i rozpakowałem je w katalogu /usr/src/linux:

```
# cd /usr/src
# wget ftp://sunsite.icm.edu.pl/pub/linux/kernel/v2.4/linux-2.4.23.tar.gz
# tar zxvf linux-2.4.23.tar.gz && mv linux-2.4.23 linux
```

Następnym krokiem było ściągnięcie patch'y do jądra, które umożliwią filtrowanie pakietów w 7-ej warstwie modelu OSI. Wszystkie niezbędne patch'e znajdziemy na stronie http://l7-filter.sf.net
Potrzebowałem trzech plików:

```
layer7-kernel2.4patch-v0.4.1a.patch
iproute2_w_layer7_patch-0.9.3.tar.gz
layer7-protocols-2004-01-02.tar.gz
```

Następnie uzupełniłem jądro:

```
# patch -p0 < layer7-kernel2.4patch-v0.4.1a.patch
patching file linux/Documentation/Configure.help
patching file linux/include/linux/netfilter_ipv4/ip_conntrack.h
patching file linux/include/linux/pkt_cls.h
patching file linux/net/ipv4/netfilter/Config.in
patching file linux/net/sched/Config.in
patching file linux/net/sched/Makefile
patching file linux/net/sched/cls_api.c
patching file linux/net/sched/cls_layer7.c
patching file linux/net/sched/regexp/regerror.c
patching file linux/net/sched/regexp/regexp.c
patching file linux/net/sched/regexp/regexp.h
patching file linux/net/sched/regexp/regmagic.h
patching file linux/net/sched/regexp/regsub.c
```

I przekompilowałem go z opcjami dostępnymi na płytce w pliku `/boot/config-2.4.23`:

```
# make dep && make bzImage && make modules && make modules_install
```

Zainstalowałem odpowiednio jądro:

```
# cp /usr/src/linux/arch/i386/boot/bzImage /boot/vmlinuz-2.4.23
# cp /usr/src/linux/.config /boot/config-2.4.23
# cp /usr/src/linux/System.map /boot/System.map-2.4.23
```

Dodałem odpowiednie wpisy do pliku `/etc/lilo.conf`, przeładowałem bootloadera poleceniem `lilo` i zrestartowałem system.
Następnie dodałem wpis do pliku `/etc/sysctl.conf`:

```
net/ipv4/ip_forward = 1
```

który umożliwi forwardowanie pakietów pomiędzy interfejsami.
Do tak przygotowanego systemu dopisałem własny skrypt uruchamiany przy starcie systemu, który umożliwia łatwe konfigurowanie zapory sieciowej.
Skrypt ten znajduje się na dysku CD w pliku:

```
/sbin/setup.sh
```

Rozpakowałem i zainstalowałem spatchowany pakiet iproute2:

```
# tar zxvf iproute2_w_layer7_patch-0.9.3.tar.gz
# cd iproute2_w_layer7_patch-0.9.3
# ./configure
# make
# make install
```

A następnie do katalogu konfiguracyjnego filtra pakietów dodałem wpisy odpowiednich reguł, dzięki którym będziemy mogli łatwo filtrować w 7-ej warstwie (dokładniejszy opis znajduje się w punkcie 5.4.):

```
# tar zxvf layer7-protocols-2004-01-02.tar.gz
# mkdir /etc/l7-proto
# find layer7-protocols-2004-01-02/ -name "*.pat" -exec cp {} /etc/l7-proto \;
```

Po zrestartowaniu na normalnym systemie podmontowałem przygotowaną wcześniej partycję z systemem filtrującym w warstwie 7-ej do miejsca `/test/`. Pozwoli nam to przygotować system operacyjny do startu z płyty CD. Napisałem skrypt `/test/etc/init.d/rc.iso` i dodałem jego wykonywanie na początku startu systemu (pierwsza linia w pliku `/test/etc/init.d/rcS`).
Następnie wykonałem następujące polecenia:

```
rm -fR /test/tmp
ln -s var/tmp /test/tmp
touch /test/proc/mounts
rm /test/etc/mtab
ln -s /proc/mounts /test/etc/mtab
mv /test/var/lib /test/lib/var-lib
mv /test/var /test/lib
mkdir /test/var
ln -s /lib/var-lib /test/lib/var/lib
rm -fR /test/lib/var/catman
rm -fR /test/lib/var/log/httpd
rm -f /test/lib/var/log/samba/*
for i in `find /test/lib/var/log -type f`; do cat /dev/null > $i; done
rm `find /test/lib/var/lock -type f`
rm `find /test/lib/var/run -type f`
```

Zmodyfikowałem odpowiednio plik `/test/etc/default/rcS` oraz `/test/etc/init.d/rcS` (oba znajdują się na płycie CD), przygotowałem plik `boot.img` celem utworzenia bootowalnej płyty CD i następnie utworzyłem obraz płyty:

```
# mkisofs -R -b boot.img -c boot.catalog -o boot.iso /test
```

Tak przygotowany obraz wypaliłem na płytce.

5.2. Iptables

Do filtrowania pakietów używam pakietu iptables. Konfiguracja iptables odbiega nieco od znanego z wcześniejszych wersji ipchains, głównie dlatego że nowy filtr jest w dużej mierze modularny. Opcje, które kiedyś były stałymi parametrami programu ipchains są obecnie realizowane przez poszczególne moduły. Do poprawnego działania potrzebne są odpowiednie moduły wkompilowane w jądro 2.4 oraz program iptables, służący do konfiguracji filtra. Filozofia nowego filtra jest dość podobna — mamy tutaj także trzy domyślne zestawy reguł INPUT, FORWARD i OUTPUT oraz szereg celów (targets), które określaja, co należy zrobić z pakietem pasującym do danej regułki. Najczęściej używane to ACCEPT, DROP (zablokowanie pakietu bez powiadomienia nadawcy) oraz REJECT (zablokowanie ze zwróceniem komunikatu ICMP). Warto jednak dodać, że w nowym filtrze zasadniczo zmieniły sie reguły przepuszczania pakietów przez poszczególne zestawy reguł. Pakiety przesyłane (forwarded) z innych interfejsów przechodzą wyłacznie przez zestaw FORWARD. Dwa pozostałe zestawy — INPUT i OUTPUT obsługują wyłącznie pakiety kończące drogę na lokalnym systemie lub na nim generowane.

Nowy filtr pakietów oferuje także pojęcie stanów. Znanymi stanami są INVALID - oznaczający, że pakiet nie jest znany i powiązany z jakimkolwiek nawiązanym już przez zaporę połączeniem, ESTABLISHED – pakiet jest częścią istniejącego aktywnego połączenia przechodzącego przez zaporę, NEW – oznacza nowe połączenie (inicjację), RELATED – oznacza, że pakiet inicjuje nowe połączenie, ale w odniesieniu do już istniejącego (np. transfer danych FTP).

W zestawie reguł do używanego przeze mnie firewall'a zastosowałem następujące reguły:

```
1.    iptables -P INPUT DROP
2.    iptables -P OUTPUT ACCEPT
3.    iptables -P FORWARD DROP
4.    iptables -I INPUT -i lo -j ACCEPT
5.    iptables -I INPUT -p tcp -m state --state RELATED,ESTABLISHED -j ACCEPT
6.    iptables -I INPUT -p udp -m state --state RELATED,ESTABLISHED -j ACCEPT
7.    iptables -I INPUT -p icmp -m state --state RELATED,ESTABLISHED -j ACCEPT
8.    iptables -I FORWARD -p tcp -m state --state RELATED,ESTABLISHED -j
```

```
   ACCEPT
9.    iptables -I FORWARD -p udp -m state --state RELATED,ESTABLISHED -j
   ACCEPT
10.   iptables -I FORWARD -p icmp -m state --state RELATED,ESTABLISHED -j
   ACCEPT
11.   iptables -I FORWARD -p tcp ! --syn -m state --state NEW -j DROP
12.   iptables -I FORWARD -p tcp -s $NWM -j ACCEPT
13.   iptables -I FORWARD -p udp -s $NWM -j ACCEPT
14.   iptables -I FORWARD -p icmp -s $NWM -j ACCEPT
15.   iptables -t nat -I POSTROUTING -p tcp -s $NWM -j MASQUERADE
```

Gdzie `$NWM` oznacza sieć wewnętrzną. Pierwsze trzy reguły ustawiają domniemane działania, następna przepuszcza ruch do interfejsu loopback. Trzy kolejne wpisy przepuszczają takie dane protokołów TCP, UDP i ICMP, które są w powiązaniu już z istniejącymi (nawiązanymi) połączeniami, ale tylko do firewalla. Pakiety forwardowane ustawiane są odpowiednio w regułach FORWARD.

Reguła nr 11 zabrania dostępu do wnętrza chronionej sieci pakietów, które nie mają ustawionej flagi SYN i stanowią inicjację połączenia. Dokładniejszy opis tego zjawiska znajdziemy na stronie http://insecure.org.

Ostatni wpis dokonuje maskowania sieci wewnętrznych. Dokładniejszy opis reguł i opcji iptables znajdziemy na stronie http://www.netfilter.org.

5.3. Tc

Na początek musimy poczynić kilka założeń. Po pierwsze, łącza mają ograniczoną przepustowość. Po drugie, przepustowości łącz w Internecie są bardzo zróżnicowane. Firewall może przyjmować dane z Ethernetu z prędkościami rzędu megabitów na sekundę i wysyłać je łączem szeregowym o przepustowości o dwa rzędy wielkości mniejszej. Większość omawianych poniżej procesów zachodzi na styku takiego szybkiego i wolnego łącza, co siłą rzeczy prowadzi do sytuacji, gdy część pakietów nie może zostać „wepchnięta” w wąskie gardło i jest gubiona. Zastosowanie w tym miejscu sterowania przepływem danych pozwala ograniczyć straty do minimum, poprawić wykorzystanie łącza, a także – w razie potrzeby – zapewnić określonym rodzajom danych pierwszeństwo.

`Tc` jest programem do ustawiania w systemie Linux przepływu danych. Za pomocą spatchowanego programu `tc` ustawiam również filtrowanie danych w warstwie 7-ej modelu OSI. Zainteresowanych odsyłam do opisu dostępnego na stronie `http://l7-filter.sf.net`.

5.4. Dane filtrowane w warstwie 7-ej modelu OSI

Zapora sieciowa zbudowana przeze mnie umożliwia wybranie filtrowanych danych w warstwie aplikacji. Mamy do wyboru następujące możliwości wyboru:

- Aim - AOL instant messenger (OSCAR i TOC), zazwyczaj działa na porcie 5190. W skład wchodzi także ruch ICQ.
- AIM web content – dane ściągane przez AOL Instant Messenger.
- Apple Juice - P2P - http://www.applejuicenet.de/
- Bearshare - klient Gnutella'i (używa stringu gnutella).
- Biff - nowe powiadamianie mail. Zazwyczaj działa na porcie 512.
- Bittorrent - http://sourceforge.net/projects/bittorrent/
- Code Red - worm który atakuje serwery Microsoft IIS.

- Counter Strike – Popularna gra.
- CVS - Concurrent Versions System.
- DHCP - Dynamic Host Configuration Protocol (RFC 1541). Zazwyczaj działa na porcie 67 (serwer) i 68 (klient). Zaznacza również BOOTP (Bootstrap Protocol (RFC 951)) w przypadku gdy używamy opcji "vendor specific options" (te opcje są standardowe dla DHCP).
- Direct Connect – udostępnianie plików P2P. http://www.neo-modus.com/. Ośrodek nasłuchuje na porcie 411. Informacje o protokole: http://wza.digitalbrains.com/DC/doc/Introduction.html. Używa wielopakietowego dopasowania!
- DNS - Domain Name System (RFC 1035).
- eDonkey2000 – udostępnianie plików P2P (http://edonkey2000.com). Używa wielopakietowego dopasowania!
- FastTrack - udostępnianie plików P2P (Kazaa, Morpheus, iMesh, Grokster, etc). Używa wielopakietowego dopasowania!
- Finger – Serwer informacji o użytkownikach. Zazwyczaj działa na porcie 79.
- Flash - Macromedia Flash.
- FTP - File Transfer Protocol (RFC 959). Zazwyczaj działa na porcie 21. Przesyłanie danych odbywa się na portach przydzielanych dynamicznie, co oznacza, że do poprawnej klasyfikacji tych pakietów będzie potrzebny moduł "FTP connection tracking" w jądrze.
- GIF - Popular Image format.
- Gkrellm – monitorowanie systemu (http://gkrellm.net).
- GnucleusLAN (wersja tylko LAN) – udostępnianie plików P2P.
- Gnutella - udostępnianie plików P2P. Tego protokołu używają różne klienty

takie jak: Mactella, Gnucleus, Gnotella, LimeWire, BearShare, iMesh i WinMX. Reguły filtrujące są rozdzielone na w.w. protokoły. Możesz je filtrować niezależnie.
- Gopher - prekursor HTTP (RFC 1436). Zazwyczaj działa na porcie 70.
- H.323 - Voice over IP. Może dopasowywać dane używające TPKT i Q.931.
- (X)HTML - (Extensible) Hypertext Markup Language - http://w3.org.
- HTTP - HyperText Transfer Protocol (RFC 2616).
- Ident - Identification Protocol - RFC 1413.
- IMAP - Internet Message Access Protocol.
- IP printing – nowy standard do drukowania UNIX (RFC 2910).
- IRC - Internet Relay Chat - RFC 1459. Zazwyczaj działa na porcie 6666 albo 6667.
- Jabber - otwarty protokół wiadomości - http://jabber.org.
- JPEG - format Joint Picture Expert Group.
- live365 - radio przez internet (http://live365.com).
- MSN Messenger - klient Microsoft Network chat. Zazwyczaj działa na porcie 1863. http://www.hypothetic.org/docs/msn/index.php.
- mySQL - baza danych.
- Netbios name service.
- NCP - Novell Core Protocol.
- NetBIOS - Network Basic Input Output System. Ten protokół jest czasem nazywany jako Common Internet File System (CIFS), LanManager albo NetBIOS. Aktulanie, SMB jest protokołem warstwy wyższej jak NetBIOS. Jakkolwiek nagłówek NetBIOS ma tylko 4 bajty – nie wiele do klasyfikacji. http://www.ubiqx.org/cifs/SMB.html. Ta reguła pasuje do warstwy sesji protokołu NetBIOS. Tutaj pakiety są klasyfikowane wcześniej jak w łatwym do zaznaczenia nagłówku protokołu SMB.
- Nimda - worm który atakuje serwery Microsoft IIS.
- NNTP - Network News Transfer Protocol (RFC 977, 2980). Zazwyczaj działa na porcie 119.
- Ogg – muzyczny format danych Ogg Vorbis (nie pliki ogg, tylko vorbis).
- PDF - Portable Document Format - Postscript-jak format stworzony przez firmę Adobe.
- Perl - język programowania stworzony przez Larry Wall.
- POP3 - Post Office Protocol version 3 (RFC 1939).
- Postscript - Printing Language.

- Pressplay - legalna strona z udostępnianą muzyką (http://pressplay.com).
- Quake 2, Quake 3, Quake World, Half Life – popularna gra.
- Quake 1 - popularna gra.
- RDP - Remote Desktop Protocol (używany w Windows Terminal Services).
- Rlogin - remote login (RFC 1282).
- RPM - Redhat Package Management.
- RTSP - Real Time Streaming Protocol - http://www.rtsp.org/. Zazwyczaj działa na porcie 554.
- RTF - Rich Text Format – otwarty format dokumentów.
- Samba/SMB - Server Message Block - udostępnianie plików "Microsoft File Sharing". Ten protokół jest czasem nazywany jako Common Internet File System (CIFS), LanManager albo NetBIOS. Aktulanie, SMB jest protokołem warstwy wyższej jak NetBIOS. Jakkolwiek nagłówek NetBIOS ma tylko 4 bajty – nie wiele do klasyfikacji. http://www.ubiqx.org/cifs/SMB.html
- SMTP - Simple Mail Transfer Protocol (RFC 2821, 1869). Zazwyczaj działa na porcie 25.
- SNMP - Simple Network Management Protocol (RFC1157). Zazwyczaj działa na porcie UDP 161 (monitoring) i 162 (traps). Ten filtr klasyfikuje pakiety SNMPv1 i nie oznacza pakietów zakodowanego protokołu ASN.1. Jakkolwiek SNMPv1 może być klasyfikowane przez inne filtry, które używają zakodowanego protokołu ASN.1.
- SNMP Monitoring - Simple Network Management Protocol (RFC1157). Zazwyczaj działa na porcie UDP 161.
- SNMP Traps - Simple Network Management Protocol (RFC1157). Zazwyczaj działa na porcie UDP 162.
- SOCKS Version 5 - protokół trawersowania dla firewall'I (RFC 1928). Zazwyczaj działa na porcie 1080. Bardzo użyteczny: http://www.iana.org/assignments/socks-methods.
- SSH - Secure SHell. Zazwyczaj działa na porcie 22.
- Tar – archiwizacja danych. Standardowy archiwizator plików dla systemów UNIX.
- Telnet – zdalne logowanie otwartym tekstem (RFC 854). Zazwyczaj działa na porcie 23.
- TFTP - Trivial File Transfer Protocol (RFC 1350). Zazwyczaj działa na porcie 69.
- Valid certificate SSL – wszystko tunelowane przez SSL (n.p.: HTTPS,

IMAPS). Ta reguła dopasowuje dane podpisane przez centrum autoryzacji.
- VNC - Virtual Network Computing. Znane też jako RFB - Remote Frame Buffer. http://www.realvnc.com/documentation.html.
- WinMX – klient Gnutella'i.
- X Windows Version 11 – sieciowy interfejs GUI używany w jednolitych systemach. Specyfikacja: http://www.msu.edu/~huntharo/xwin/docs/xwindows/PROTO.pdf Zazwyczaj działa na porcie 6000 (dla drugiego serwera na hoście 6001, itd.).
- Yahoo messenger – protokół przysyłania wiadomości (http://yahoo.com). Zazwyczaj działa na porcie 5050.

Przy wyborze odpowiedniej opcji dane trafiają do odpowiedniej reguły utworzonej przez filtr `tc`. Tutaj za pomocą QoS ograniczamy prędkość odpowiednich danych.

5.5. Podsumowanie

Linuxowa implementacja algorytmów QoS nie jest, jak się łatwo domyślić, jedyna w świecie ruterów. Wiele innych systemów posiada zaimplementowane różne algorytmy kolejkujące oraz mechanizmy QoS. Cisco IOS posiada implementacje algorytmów WFQ (*Weighted Fair Queueing*), PQ (*Priority Queueing*), WRED (*Weighted Random Early Detection*) oraz CBQ, przy czym to ostatnie określane jest jako custom queueing. IOS posiada także możliwość przycinania pasma (*traffic-shaping*) dostępnego dla danych, pasujących do określonych reguł. Dla rodziny systemów BSD (FreeBSD, OpenBSD oraz NetBSD) jest dostępny pakiet ALTQ, zawierający implementacje algorytmów CBQ, RED, WFQ i innych. Na stronach obu wymienionych implementacji można znaleźć bardzo dużo ogólnych informacji na temat działania algorytmów QoS, przydatnych także dla użytkowników Linuxa.

Mnie zainteresował `tc` z racji elastyczności, poniesionych nakładów finansowych a także z powodu ciągłego szybkiego rozwoju.

6. Analiza i testy zabezpieczonej sieci

W tym rozdziale opisano testy działania wykonanej zapory sieciowej. Omówiono je od strony zabezpieczonej sieci, od strony zewnętrznej jak i zanalizowano wewnętrzne problemy mogące się pojawić w czasie pracy systemu. Celem testów nie jest sprawdzenie zapory pod względem wydajności (za bardzo jest to zależne od sprzętu) lecz pod względem funkcjonalności i efektywności.

6.1. Prawidłowość filtrowania danych

Pierwszy test ma na celu sprawdzenie, czy zapora sieciowa w ogóle przepuszcza jakikolwiek ruch z wnętrza zabezpieczonej sieci.

W tym celu startujemy przygotowany system i odpowiednio go konfigurujemy. Komputer, na jakim dokonuje testów posiada dwie karty sieciowe RealTek 8139. Jedna z kart ma nadany nr IP wewnętrzny 192.168.1.1/24, druga zaś posiada nr IP zewnętrzny 192.168.7.78/24. Komputer w sieci wewnętrznej to Windows 98 z ustawionym numerem IP 192.168.1.2/24 i domyślną bramą 192.168.1.1. Komputer z systemem Windows 98 jest podłączony bezpośrednio skrosowanym przewodem z firewallem, ten zaś poprzez switch jest wpięty do sieci zewnętrznej. Tak wygląda konfiguracja i stan interfejsów po uruchomieniu firewall'a:

```
# ip address list
1: lo: <LOOPBACK> mtu 16436 qdisc noop
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc htb qlen 1000
    link/ether 00:c0:26:a5:c5:9f brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.1/24 brd 192.168.1.255 scope global eth0
3: eth1: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc htb qlen 1000
    link/ether 00:00:21:ec:58:e0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.7.78/24 brd 192.168.7.255 scope global eth1

# iptables -L -n -v
Chain INPUT (policy DROP 4 packets, 755 bytes)
 pkts bytes target     prot opt in     out     source
destination
    2   168 ACCEPT     icmp --  *      *       0.0.0.0/0           0.0.0.0/0
state RELATED,ESTABLISHED
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0           0.0.0.0/0
state RELATED,ESTABLISHED
   22  2992 ACCEPT     tcp  --  *      *       0.0.0.0/0           0.0.0.0/0
state RELATED,ESTABLISHED
    0     0 ACCEPT     all  --  lo     *       0.0.0.0/0           0.0.0.0/0
```

```
Chain FORWARD (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source
destination
    0     0 ACCEPT     icmp --  *      *       192.168.1.0/24       0.0.0.0/0
    0     0 ACCEPT     udp  --  *      *       192.168.1.0/24       0.0.0.0/0
    0     0 ACCEPT     tcp  --  *      *       192.168.1.0/24       0.0.0.0/0
    0     0 DROP       tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
tcp flags:!0x16/0x02 state NEW
    0     0 ACCEPT     icmp --  *      *       0.0.0.0/0            0.0.0.0/0
state RELATED,ESTABLISHED
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0
state RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0
state RELATED,ESTABLISHED

Chain OUTPUT (policy ACCEPT 33 packets, 4179 bytes)
 pkts bytes target     prot opt in     out     source
destination

# iptables -t nat -L -n -v
Chain PREROUTING (policy ACCEPT 3 packets, 518 bytes)
 pkts bytes target     prot opt in     out     source
destination

Chain POSTROUTING (policy ACCEPT 5 packets, 392 bytes)
 pkts bytes target     prot opt in     out     source
destination
    0     0 MASQUERADE  tcp  --  *      *       192.168.1.0/24       0.0.0.0/0

Chain OUTPUT (policy ACCEPT 5 packets, 392 bytes)
 pkts bytes target     prot opt in     out     source
destination

# tc -s qdisc ls dev eth0
qdisc sfq 10: quantum 1514b perturb 10sec
 Sent 4207 bytes 32 pkts (dropped 0, overlimits 0)

 qdisc htb 1: r2q 10 default 10 direct_packets_stat 0
 Sent 4207 bytes 32 pkts (dropped 0, overlimits 0)

# tc -s qdisc ls dev eth1
qdisc sfq 10: quantum 1514b perturb 10sec
 Sent 378 bytes 5 pkts (dropped 0, overlimits 0)

 qdisc htb 1: r2q 10 default 10 direct_packets_stat 0
 Sent 378 bytes 5 pkts (dropped 0, overlimits 0)

# tc -s class ls dev eth0
class htb 1:1 root rate 1Mbit ceil 1Mbit burst 15Kb cburst 2909b
 Sent 4207 bytes 32 pkts (dropped 0, overlimits 0)
 lended: 0 borrowed: 0 giants: 0
 tokens: 94552 ctokens: 16740
```

```
class htb 1:10 parent 1:1 leaf 10: prio 0 rate 1Mbit ceil 1Mbit burst 15Kb
cburst 2909b
 Sent 4207 bytes 32 pkts (dropped 0, overlimits 0)
 lended: 32 borrowed: 0 giants: 0
 tokens: 94552 ctokens: 16740

# tc -s class ls dev eth1
class htb 1:1 root rate 1Mbit ceil 1Mbit burst 15Kb cburst 2909b
 Sent 378 bytes 5 pkts (dropped 0, overlimits 0)
 lended: 0 borrowed: 0 giants: 0
 tokens: 95500 ctokens: 17688

class htb 1:10 parent 1:1 leaf 10: prio 0 rate 1Mbit ceil 1Mbit burst 15Kb
cburst 2909b
 Sent 378 bytes 5 pkts (dropped 0, overlimits 0)
 lended: 5 borrowed: 0 giants: 0
 tokens: 95500 ctokens: 17688

# tc -s filter ls dev eth0
# tc -s filter ls dev eth1
```

Jak możemy zauważyć komputer z wnętrza sieci łączył się już z internetem (świadczą o tym liczniki iptables oraz ruch klasyfikowany odpowiednio na kolejkach QoS interfejsów). Nazwy domenowe rozwiązują się prawidłowo (ruch UDP), strony WWW działają bez zarzutu (TCP), pingi również działają prawidłowo (ICMP). Telnet na port 25 (SMTP) serwera z internetu również pokazuje, że wszystko działa jak należy. Liczniki na firewallu zwiększają się odpowiednio wraz z ilością ściągniętych z internetu danych.

Drugi test ma na celu pokazanie skuteczności filtrowania danych w warstwie 7-ej modelu OSI. W tym celu użyję najczęściej spotykanego klienta Kazaa.

Restartuję firewall'a i wpisuję konfigurację jak wyżej, zaznaczając tym razem, że chcę filtrować w warstwie 7-ej. Wśród dostępnych danych zaznaczam tylko opcję „FastTrack" (opis opcji dostępny jest w punkcie 5.4).

Sprawdzam liczniki przed próbą ściągnięcia danych:

```
# tc -s qdisc ls dev eth0
qdisc sfq 11: quantum 1514b perturb 10sec
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)

 qdisc sfq 10: quantum 1514b perturb 10sec
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)

 qdisc htb 1: r2q 10 default 10 direct_packets_stat 0
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
```

```
# tc -s qdisc ls dev eth1
qdisc sfq 12: quantum 1514b perturb 10sec
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)

 qdisc sfq 10: quantum 1514b perturb 10sec
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)

 qdisc htb 1: r2q 10 default 10 direct_packets_stat 0
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)

# tc -s class ls dev eth0
class htb 1:11 parent 1:1 leaf 11: prio 0 rate 360Kbit ceil 360Kbit burst
1023b cburst 2Kb
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
 lended: 0 borrowed: 0 giants: 0
 tokens: 18204 ctokens: 36621

class htb 1:1 root rate 1Mbit ceil 1Mbit burst 15Kb cburst 2909b
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
 lended: 0 borrowed: 0 giants: 0
 tokens: 95999 ctokens: 18187

class htb 1:10 parent 1:1 leaf 10: prio 0 rate 1Mbit ceil 1Mbit burst 15Kb
cburst 2909b
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
 lended: 0 borrowed: 0 giants: 0
 tokens: 95999 ctokens: 18187

# tc -s class ls dev eth1
class htb 1:1 root rate 1Mbit ceil 1Mbit burst 15Kb cburst 2909b
 Sent 42 bytes 1 pkts (dropped 0, overlimits 0)
 rate 1bps
 lended: 0 borrowed: 0 giants: 0
 tokens: 95750 ctokens: 17938

class htb 1:10 parent 1:1 leaf 10: prio 0 rate 1Mbit ceil 1Mbit burst 15Kb
cburst 2909b
 Sent 42 bytes 1 pkts (dropped 0, overlimits 0)
 rate 1bps
 lended: 1 borrowed: 0 giants: 0
 tokens: 95750 ctokens: 17938

class htb 1:12 parent 1:1 leaf 12: prio 0 rate 360Kbit ceil 360Kbit burst
1023b cburst 2Kb
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)
 lended: 0 borrowed: 0 giants: 0
 tokens: 18204 ctokens: 36621

# tc -s filter ls dev eth0
filter parent 1: protocol ip pref 1 layer7

# tc -s filter ls dev eth1
filter parent 1: protocol ip pref 1 layer7
```

Jak można łatwo zauważyć pojawiły sie dodatkowe klasy.
Dla eth0:

```
class htb 1:11 parent 1:1 leaf 11: prio 0 rate 360Kbit ceil 360Kbit burst
1023b cburst 2Kb
```

Dla eth1:

```
class htb 1:12 parent 1:1 leaf 12: prio 0 rate 360Kbit ceil 360Kbit burst
1023b cburst 2Kb
```

które zapewniają odpowiednie obsługiwanie sklasyfikowanych pakietów należących do aplikacji Kazaa. Następnie uruchamiam program, a następnie obserwuję stany kolejek:

```
# tc -s qdisc ls dev eth0
qdisc sfq 11: quantum 1514b perturb 10sec
 Sent 316817 bytes 366 pkts (dropped 0, overlimits 0)

 qdisc sfq 10: quantum 1514b perturb 10sec
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)

 qdisc htb 1: r2q 10 default 10 direct_packets_stat 0
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)

# tc -s qdisc ls dev eth1
qdisc sfq 12: quantum 1514b perturb 10sec
 Sent 4153 bytes 36 pkts (dropped 0, overlimits 0)

 qdisc sfq 10: quantum 1514b perturb 10sec
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)

 qdisc htb 1: r2q 10 default 10 direct_packets_stat 0
 Sent 0 bytes 0 pkts (dropped 0, overlimits 0)


# tc -s class ls dev eth1
class htb 1:1 root rate 1Mbit ceil 1Mbit burst 15Kb cburst 2909b
 Sent 43845 bytes 448 pkts (dropped 0, overlimits 0)
 rate 3bps
 lended: 0 borrowed: 0 giants: 0
 tokens: 95750 ctokens: 17938

class htb 1:10 parent 1:1 leaf 10: prio 0 rate 1Mbit ceil 1Mbit burst 15Kb
cburst 2909b
 Sent 39638 bytes 411 pkts (dropped 0, overlimits 0)
 rate 1bps
 lended: 411 borrowed: 0 giants: 0
 tokens: 95750 ctokens: 17938
```

```
class htb 1:12 parent 1:1 leaf 12: prio 0 rate 360Kbit ceil 360Kbit burst
1023b cburst 2Kb
 Sent 4207 bytes 37 pkts (dropped 0, overlimits 0)
 rate 1bps
 lended: 37 borrowed: 0 giants: 0
 tokens: 17352 ctokens: 35769
```

Jak możemy zauważyć pakiety są prawidłowo klasyfikowane do odpowiednich kolejek. Ruch aplikacji Kazaa zostaje prawidłowo ograniczony.

6.2. Nmap

Aby stwierdzić, że firewall prawidłowo pracuje i spełnia powierzone mu zadanie, musimy również sprawdzić, że broni on sieć wewnętrzną przed atakami z zewnątrz. W celu sprawdzenia samego firewalla posłużę się skanerem portów `Nmap` (http://www.insecure.org/nmap/).

Oto wynik badań ze stacji będącej po stronie internetu (ustawienia firewall'a pozostawiam niezmienione z poprzedniego badania – mam w ten sposób pewność, że jest on w trakcie wykonywania swojej pracy).

```
# nmap -sS -O 192.168.7.78/32

Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Note: Host seems down. If it is really up, but blocking our ping probes, try -
P0

Nmap run completed -- 1 IP address (0 hosts up) scanned in 30 seconds

# nmap -sU 192.168.7.78/32

Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Note: Host seems down. If it is really up, but blocking our ping probes, try -
P0

Nmap run completed -- 1 IP address (0 hosts up) scanned in 30 seconds

# nmap -sF 192.168.7.78/32

Starting nmap V. 2.54BETA31 ( www.insecure.org/nmap/ )
Note: Host seems down. If it is really up, but blocking our ping probes, try -
P0

Nmap run completed -- 1 IP address (0 hosts up) scanned in 30 seconds
```

Jak widzimy żaden pakiet nie dotarł do firewall'a. Nic dziwnego, skoro firewall nie ma otwartych żadnych portów a przy tym odpowiednie pakiety są kierowane do

reguły DROP. Później zobaczymy, że trochę inne narzędzie, jakim jest `hping` również miało kłopoty ze zdiagnozowaniem przeciwnika.

6.3. Hping

`Hping` to narzędzie zorintowane na tworzenie różnych pakietów (http://www.hping.org). Możemy kształtować nagłówki dowolnie i je następnie „wpychać" do sieci obserwując jej zachowanie. Za pomocą tego narzędzia zbadam zabezpieczenie zapory i jej skuteczność w przepuszczaniu pakietów do wnętrza chronionej sieci.

Na początku ustawiam w tablicy rutowania komputera z sieci wewnętrznej, aby pakiety z adresem docelowym 192.168.1.0/24 kierował przez badanego firewall'a.

```
# ip r a 192.168.1.0/24 via 192.168.7.78
```

Teraz spróbuję zwykłego ping'a do maszyny z MS W98 wykonując go z sieci zewnętrznej:

```
# ping -c 3 192.168.1.2
PING 192.168.1.2 (192.168.1.2): 56 data bytes

--- 192.168.1.2 ping statistics ---
3 packets transmitted, 0 packets received, 100% packet loss
```

Pakiety ICMP zostały odrzucone. Oznacza to o prawidłowym filtrowaniu danych ICMP.
Teraz spróbujemy czegoś innego. Pakiety, które nie mają ustawionej flagi SYN i inicjują połączenie normalnie nie powinny się pojawiać w sieci. Możemy więc zainicjować skanowanie pakietami ACK. Najpierw spróbujemy przeskanować firewall na różne porty:

```
# hping 192.168.7.78 -A -p 22 -c 3
HPING 192.168.7.78 (eth0 192.168.7.78): A set, 40 headers + 0 data bytes

--- 192.168.7.78 hping statistic ---
3 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

# hping 192.168.7.78 -A -p 80 -c 3
HPING 192.168.7.78 (eth0 192.168.7.78): A set, 40 headers + 0 data bytes
```

```
--- 192.168.7.78 hping statistic ---
3 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Następnie sprawdzamy, jakie porty ma pootwierane MS W98:

```
Microsoft(R) Windows 98
   (C)Copyright Microsoft Corp 1981-1999.

C:\WINDOWS\Pulpit>netstat -an

Aktywne połączenia

  Protokół  Adres lokalny          Obcy adres        Stan
  TCP    0.0.0.0:1115           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1127           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1139           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1143           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1149           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1150           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1151           0.0.0.0:0              LISTENING
  TCP    127.0.0.1:1063         0.0.0.0:0              LISTENING
  TCP    127.0.0.1:1076         0.0.0.0:0              LISTENING
  TCP    192.168.1.2:1139       192.168.7.222:22       ESTABLISHED
  TCP    192.168.1.2:137        0.0.0.0:0              LISTENING
  TCP    192.168.1.2:138        0.0.0.0:0              LISTENING
  TCP    192.168.1.2:139        0.0.0.0:0              LISTENING
  UDP    127.0.0.1:1063         *:*
  UDP    127.0.0.1:1076         *:*
  UDP    192.168.1.2:137        *:*
  UDP    192.168.1.2:138        *:*

C:\WINDOWS\Pulpit>
```

I próbujemy skanowania na jego pootwierane porty:

```
# hping 192.168.1.2 -A -p 1115 -c 3
HPING 192.168.1.2 (eth0 192.168.1.2): A set, 40 headers + 0 data bytes

--- 192.168.1.2 hping statistic ---
3 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

# hping 192.168.1.2 -A -p 138 -c 3
HPING 192.168.1.2 (eth0 192.168.1.2): A set, 40 headers + 0 data bytes

--- 192.168.1.2 hping statistic ---
3 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

```
# hping 192.168.1.2 -2 -p 137 -c 3
HPING 192.168.1.2 (eth0 192.168.1.2): udp mode set, 28 headers + 0 data bytes

--- 192.168.1.2 hping statistic ---
3 packets tramitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

W ten sposób sprawdziliśmy zaporę sieciową, co do bezpieczeństwa filtrowania danych TCP i UDP. Wszystkie testy wypadły pomyślnie.

6.4. Podsumowanie

W przeprowadzonych testach pokazałem, że firewall stworzony przeze mnie prawidłowo umożliwia użytkownikom dostęp do zasobów internetu, prawidłowo funkcjonuje i klasyfikuje odpowiednie dane. Próby skanowania zapory sieciowej nie przynoszą żadnych rezultatów. Podobnie próby skanowania sieci zabezpieczonej przez firewall. Wszystkie testy wypadły pomyślnie.

7. Podsumowanie

Celem pracy było stworzenie bezpiecznej zapory sieciowej filtrującej w warstwie aplikacji i posiadającej mechanizmy kontroli przepływu - QoS.

W pracy tej zawarłem również opis modelu OSI, do którego powinna zajrzeć każda osoba zaczynająca swoją pracę z sieciami.

Zapora sieciowa jest przeznaczona zarówno dla użytkowników mających małe doświadczenie w administrowaniu sieciami komputerowymi (z racji łatwego interfejsu konfiguracji) jak i dla bardziej zaawansowanych z racji dostępnych możliwości filtrowania oraz możliwości tworzenia własnych konfiguracji (przy pierwszym starcie systemu zapisujemy konfigurację na dyskietce, a następnie zapisujemy na niej własne ustawienia do pliku config.sh – dyskietka posiada system plików ext2).

Z pewnością największą zaletą tej zapory sieciowej jest możliwość filtrowania w 7-ej warstwie. Wśród dostępnych rozwiązań nie spotkałem jeszcze podobnego systemu z możliwością startu z płyty CD.

Wydajność zapory przy filtrowaniu w wartwie 7-ej znacznie spada i jest silnie uzależniona od dostępnego sprzętu na jakim jest zainstalowana, jednak możemy nie zaznaczać określonej opcji przy starcie (bądź w pliku konfiguracyjnym) przez co uzyskamy system z filtrowaniem do warstwy 4-ej i możliwościami jak zwykły system Linux [21].

Dokonałem testów przepustowości opisywanej zapory. Do tego celu użyłem komputera PC z procesorem Pentium 166MHz i 12MB pamięci RAM. Zapora była bezpośrednio połączona do jednego z interfejsów z komputerem, z którego dokonywałem testów, natomiast poprzez HUB z drugim (serwerem). Przez zaporę ściągałem dane o wielkości 121MB poprzez protokół HTTP. Bez filtrowania w warstwie 7-ej (bez potwierdzania odpowiedniej opcji przy starcie zapory) prędkość ściągania danych wynosiła 6,32 MB/s. Przy włączonej opcji „Layer7” prędkość zmalała do 1,19 MB/s.

Nie używałem jeszcze swojego firewall’a w sieciach komercyjnych z racji wczesnego etapu rozwoju. Przeprowadzałem jednak próby w środowisku 30-tu stacji roboczych przy utworzeniu swojego pliku konfiguracyjnego na dyskietce. System zachowywał się stabilnie przez 29-ięć godzin, niestety musiałem powrócić do standardowego rozwiązania z racji posiadania Proxy.

W internecie można znaleźć kilka podobnych rozwiązań – niestety większość z nich została przekształcona na produkty komercyjne. Jednym z nich jest Gibraltar [19]. Zainteresowanych zachęcam do szczegółowej analizy tej zapory sieciowej – znajduje się w niej wiele ciekawych rozwiązań, bazują jednak one w znacznej części na rozwiązaniach typu Open Source.

Firewall jest przetestowany i spełnia wszystkie wymogi, jakie były postawione w założeniu. Kolejne wersje firewall'a będą się ukazywały na stronie: http://www.bmk-it.com/projects/layer7-firewall/. W planach mam dodanie serwera DNS i Proxy – wymagany będzie podłączony do komputera dysk. Dane będą filtrowane i obsługiwane przez firewall w sposób przeźroczysty (nie zauważalny dla użytkownika).

Bibliografia

1. Albitz P. & Cricket L.: *DNS i BIND.* RM, Warszawa 2001.
2. Braden R.: *Requirements for Internet Hosts - Communication Layers*. RFC 1122, Oct. 1989.
3. Cheswick W. R., Bellovin S. M.: *Firewalls and Internet Security: Repelling the Wily Hacker.* Reading, Mass., Addison-Wesley 1994.
4. Garfinkel S. L., Spafford E. H.: *Practical UNIX and Internet Security.* Wyd. 2, Sebastopol Calif., O'Reilly & Associates 1996.
5. Kent S. T.: *U.S. Department of Defense Security Options for the Internet Protocol.* RFC 1108, Niev. 1991.
6. Kernighan B. W., Ritchie D. M.: *Język ANSI C.* WNT, Warszawa 2000.
7. Muffet A.: *FAQ: Computer Security Frequently Asked Questions.* Usenet, alt.security, comp.security.misc, comp.security.unix, news.answers, Dec. 1993.
8. Postel J. B.: *Internet Control Message Protocol.* RFC 792, Sept. 1981.
9. Postel J. B.: *Internet Protocol.* RFC 791, Sept. 1981.
10. Postel J. B.: *Transmission Control Protocol.* RFC 793, Sept. 1981.
11. Postel J. B.: *User Datagram Protocol.* RFC 768, Aug. 1980.
12. Schimmel C.: *UNIX Systems for Modern Architectures: Symmetric Multiprocessing and Caching for Kernel Programmers.* Reading, Mass., Addison-Wesley 1994.
13. Stevens R. W.: *Advanced Programming in the UNIX Environment.* Reading, Mass., Addison-Wesley 1994.
14. Stevens R. W.: *TCP/IP Illustrated, Volume 1: The Protocols.* Reading, Mass., Addison-Wesley 1994.
15. Stevens R. W.: *UNIX programowanie usług sieciowych.* WNT, Warszawa 2000.
16. Praca zbiorowa, www.debian.org
17. Praca zbiorowa, www.firewallguide.com
18. Praca zbiorowa, www.freeswan.org
19. Rene Mayrhofer, www.gibraltar.at
20. Praca zbiorowa, www.insecure.org
21. Praca zbiorowa, www.iptables.org

22. Ethan Sommer i Matthew Strait: http://l7-filter.sf.net
23. Praca zbiorowa, www.stonesoft.com
24. Patton S., Doss D., Yurcik W.: *Open Source Versus Commercial Firewalls: Functional Comparison.* Annual IEEE Conference on Local Computer Networks (LCN 2000), Tampa Fl., USA, Niev. 2000.