

Państwowa Wyższa Szkoła Zawodowa
w Jeleniej Górze „KOLEGIUM KARKONOSKIE”

Institut Techniki wydział

Elektrotechniki i telekomunikacji specjalizacja Inżynieria komputerowa



Praca Dyplomowa

Konfiguracja Firewalla w systemach uniksowych

Daniel Wasyliszyn

Pod Kierunkiem

dr Andrzej Dudek

Podziękowania

Pragnę podziękować panu dr Andrzejowi Dudkowi oraz mgr.inż Markowi Szycowi, których rady i wskazówki pomogły mi rozwiązać wiele problemów podczas pisania pracy.

Spis Treści:

I. Usługi sieciowe wymagające zabezpieczeń	6
1. Bezpieczeństwo sieci	6
2. Sposoby ochrony sieci	7
3. Wady firewalli	11
4. Usługi internetowe	12
5. WWW	12
6. Poczta elektroniczna	13
7. Grupy dyskusyjne	14
8. Transfer plików	14
9. Współdzielenie plików	15
10. Drukowanie	16
11. Zdalny dostęp	16
12. Zdalne interfejsy graficzne	16
13. Usługi konferencyjne czasu rzeczywistego	17
14. Usługi nazewnicze i katalogowe	17
15. Strategie zabezpieczeń	18
II. Budowa ścian ogniowych	19
III. Architektura firewalli	26
3.1 Ruter osłaniający	26
3.2 Host dwusieciowy	27
3.3 Rozwiązania wielofunkcyjne	28
3.4 Architektura ekranowanego hosta	28
3.5 Architektura ekranowanej podsieci	30
3.6 Architektury z wieloma ekranowanymi podsieciami	32
3.7 Wieloczęściowa sieć ekranowana (split-screened subnet)	32
3.8 Niezależne ekranowane podsieci (independent screened subnet)	34
IV. Ataki oparte na szczegółach niskopoziomowych protokołów	37

4.1 Skanowanie portów.....	37
4.2 Słabość implementacji	38
4.3 Podszywanie się.....	38
4.4 Przechwytywanie pakietów	41
V. Technologie stosowane w firewallach.....	42
5.1 Flirty pakietów	43
5.2 Dynamiczne (stanowe) filtrowanie pakietów	44
5.3 Filtrowanie na podstawie adresu.....	44
5.4 Maskowanie adresu IP	45
5.5 Proxy	46
5.6 Szyfrowane tunele.....	47
5.7 Szyfrowane uwierzytelnianie.....	47
VI. Systemy Uniksowe	48
6.1 IP-Filter.....	49
6.1.1 NAT i proxy.....	54
6.1.2 Zmiany reguł IP-Filtra	57
6.1.3 Zmiany reguł IP-Filtra dla NAT	57
6.1.4 Dodatkowe narzędzia dla IP-Filter	58
6.2. Firewall oparty na systemie operacyjnym LINUXS.....	65
6.2.1 Przygotowanie Linuksa.....	65
6.2.2 Filtrowanie Protokołu ICMP	80
6.2.3 Filtrowanie na pakietów z ustawioną flagą.....	81
6.2.4 Obsługa fragmentacji.....	81
6.2.5 Zmiana Typ Usługi (ToS).....	83
6.2.6 Rozszerzenia UDP	86
6.2.7 Rozszerzenia ICMP	87
6.2.8 Test Stanu	87
6.3 NAT w Linuksie	89
6.3.1 NAT dla jądra 2.4	93
6.3.2 DHCP a firewall.....	96

VII. Proxy	97
7.1 Serwery proxy działające na poziomie aplikacji i na poziomie obwodu.....	97
7.2 Inteligentne serwery proxy.	97
7.3 SOCKS.....	97
7.4 TIS (Internet Firewall Toolkit)	98
7.5 SQUID	100
VIII. Załączniki.....	100
Przykładowe pliki konfiguracyjne	101
Bibliografia :.....	120

Cele

Celem tej pracy jest przedstawienie zasady działania i budowy zabezpieczeń sieci komputerowych opartych na ścianach ogniowych zwanych firewallami.

I. Usługi sieciowe wymagające zabezpieczeń

„Ściany ogniowe należą do najnowszych osiągnięć w technologii sieciowej. Najbardziej interesujące i innowacyjne rozwiązania jest to:

- Tłumaczenie adresów sieciowych NAT
- Filtrowanie wielowarstwowe (protokołów, pakietów itp.)
- Proxy”¹

Zanim nastąpiła era Internetu problemy bezpieczeństwa rozwiązywano za pomocą prostych filtrów pakietów i zestawu modemów typu dial-back. Jednakże szybki rozwój tego medium spowodował, że zaistniała konieczność izolowania sieci przez urządzenia, które kontrolowałyby dostęp do danych segmentów sieci i uniemożliwiałyby dostęp do niej osobą niepowołanym. Od tego momentu zaczęto tworzyć urządzenia zwane firewallami.

Aby dogłębnie zapoznać się z Firewallami musimy przeanalizować kolejne sposoby tworzenia sieci i sposoby kontrolowania do niej dostępu.

1. Bezpieczeństwo sieci

„Musimy sobie odpowiedzieć na pytanie, co musimy chronić:

- a) Dane
- b) Zasoby

¹ Firewalls Ściany ogniowe wyd. Mikom str.15

c) Reputacje (dobre imię firmy)

Musimy także dokładnie wiedzieć, z czym mamy do czynienia. Ogólnie spotykamy się z następującymi rodzajami ataków.

1. Włamania
2. Blokada usług (DoS)
3. Kradzież informacji²

Ad 1)

Są to najczęstsze rodzaje ataków i czasami nie jesteśmy świadomi faktu, że ktoś miał nieautoryzowany dostęp do naszego komputera.

Ad 2)

Ich główne zadanie to przeszkodzić w używaniu komputerów a czasami uniemożliwić korzystanie z nich na dłuższy czas. Są to jedne z najbardziej niebezpiecznych ataków powodują największe szkody. Jako przykład mogę przytoczyć atak na serwis Yahoo.

Ad 3)

Podśluchiwanie sieci czy kradzież danych czasami nie musi być poparta szeroką wiedzą wystarczy czasami telefon i osoba podająca się za osobę z kierownictwa. Do pozyskania takiej informacji wystarczy zwykła przeglądarka.

2.Sposoby ochrony sieci

Najlepszy sposób zabezpieczenia sieci to odłączenie jej od sieci zewnętrznych, ale czasami możemy zostać zaatakowani z wewnątrz naszej sieci. Dlatego musimy odpowiedzieć sobie na podstawowe pytanie.

Komu możemy zaufać?.

² Internet Firewalls wyd. RM str. 4

„Zabezpieczenie przez utajnienie

Ten model bezpieczeństwa uważany jest za najlepszy, ponieważ możemy przypuszczać, że nikt o nim nie wie (uruchomić system bezpieczeństwa nie informując nikogo o tym) a co ważniejsze nie jest znana architektura sposób stopień zabezpieczeń jednakże sposób ten raczej uważny jest niezbyt dobry im bardziej bezpieczny system tym trudniejszy w obsłudze.”³

Zabezpieczanie hosta

„Jest to jeden z najbardziej znanych sposobów zabezpieczenia polega na tym, iż każdy host zabezpiecza się osobno. Jedną z podstawowych wad tego rozwiązania to niemożliwość konfiguracji przy większej liczbie maszyn. Drugą wadą to złożoność i różnorodność systemów informatycznych. Dodatkowo musimy polegać na umiejętnościach wszystkich, którzy mają dostęp do danego hosta. Model ten odpowiedni jest dla małych firm lub, jeżeli zależy nam na wysokiej jakości zabezpieczeń.”⁴

Zabezpieczenie sieci

„Z powodu rosnącej wielkości i różnorodności środowisk informatycznych coraz częściej spotykamy ten właśnie model zabezpieczenia. W takim wypadku zabezpieczenie każdego hosta w sieci staje się coraz trudniejsze a czasami niemożliwe. Model ten polega na tworzeniu firewalli do ochrony systemów i sieci, używanie dobrych mechanizmów uwierzytelniających a także szyfrowania ważnych danych do ochrony ich podczas transmisji.”⁵

³ Internet Firewalls wyd. RM str 4

⁴Internet Firewalls wyd. RM str. 16

⁵ Internet Firewalls wyd. RM str. 17

Podsumowując **ŻADEN** z tych modeli bezpieczeństwa nie jest doskonały i nie zabezpieczy naszej sieci, danych itp. Jako przykład możemy wziąć użytkownika, który posiadając odpowiednie uprawnienia sam swoimi działaniami może spowodować świadomie lub też nie uszkodzenie naszego systemu np. ściągnięcie z Internetu czy też pocztą wirusa lub „konia trojańskiego”. Musimy zatem wdrożyć odpowiednią politykę bezpieczeństwa.

Co to jest firewall?

W naszym codziennym życiu ściana ogniowa (ang. firewall) ma za zadanie powstrzymać rozprzestrzenianie się ognia. Podobną zasadą kierujemy się przy konstruowaniu systemów zabezpieczeń komputerowych tyle tylko, że firewall bardziej obrazowo mówiąc przypomina fosę w średniowiecznym zamku z zwodzonym mostem.

„Jego podstawowe zadania to:

- Zmusza ludzi, aby wchodzili do systemu w starannie kontrolowanym punkcie.
- Zapobiega zbliżeniu się napastników do innych punktów obrony.
- Zmusza ludzi do opuszczania systemu w starannie kontrolowanym punkcie.”⁶

Fizyczna implementacja firewalle może być różna od dedykowanych firewalle (Firewall –1 firmy Checkpoint) po stary komputer z procesorem 486 z zainstalowanym oprogramowaniem ściany ogniowej. Musimy sobie jednak jasno powiedzieć, że ściany ogniowe nie zabezpieczą nas przed wszystkimi zagrożeniami. Po drugie firewalle nakładają na nas pewne ograniczenia a to nie zawsze podoba się pozostałym użytkownikom. Firewalle mają dużo zalet, ale nie zastąpią czasami prostego myślenia tych , którzy są użytkownikami danej sieci.

⁶ Internet Firewalls wyd. RM str. 19

Co potrafią firewalle?

Przede wszystkim mogą bardzo zwiększyć bezpieczeństwo sieci. Musimy myśleć o firewallu jako o najważszym miejscu. Cały ruch wchodzący i wychodzący musi przejść przez ten pojedynczy punkt, w którym zarazem koncentrują się wszystkie zabezpieczenia. Po drugie firewalle mogą egzekwować politykę bezpieczeństwa coś w rodzaju policjanta dodatkowo może on rejestrować cały ruch przechodzący przez niego.

Czego firewalle nie potrafią?

- Przede wszystkim firewalle nie rozwiążą naszych wszystkich problemów np. (nie dadzą nam ochrony fizycznej, zabezpieczenia hostów itp.).

- Firewall nie chroni przed użytkownikami wewnętrznymi, może uniemożliwić wysłanie informacji na zewnątrz za pomocą połączenia sieciowego. Ale ten sam użytkownik może skopiować dane np. na dyskietkę i wynieść je na zewnątrz.

- Firewall nie chroni przed połączeniami, które przez niego nie przechodzą. np. (Jeżeli ktoś połączy się z Internetem za pomocą modemu)

- „Firewalle nie chronią całkowicie przed wirusami. Co prawda istnieją takie ściany ogniowe, które zapewniają ochronę przed wirusami, ale jest to trudne z powodu tego że:

1. Wymaga rozpoznania, że pakiet stanowi część programu
2. Określenia jak program powinien wyglądać
3. Wykrycia, że zmiana programu jest spowodowana wirusem⁷

- Firewalle nie potrafią same się konfigurować. Nie jest możliwe podłączenie firewalle do sieci, który sam dostosuje się do danego środowiska sieciowego czy platformy.

⁷ Internet Firewalls wyd. RM str 23

3. Wady firewalli

1. „Zakłócają współpracę z Internetem wprowadzając całe mnóstwo problemów przeszkadzając użytkownikom i opóźniając wprowadzenie nowych usług internetowych.
2. Problemy, z którymi sobie nie radzą (zagrożenia wewnętrzne i połączenia przechodzące poza firewallem), są znacznie poważniejsze niż te, z którymi sobie radzą.”⁸

Ad.1)

Internet jest oparty na modelu komunikacji host – host. Firewalli zakłócają taką komunikację na wszelkie sposoby. Społeczność Internetu przyzwyczajona jest do swobodnego przepływu informacji ponadto istnieje wiele udogodnień, nie mających wpływu na bezpieczeństwo, których rozpowszechnieniu się przeszkadzają firewalli.

Ad. 2)

„Można się spotkać z opinią, że firewalli należą do przeszłości, ponieważ nie radzą sobie z prawdziwymi problemami. Musimy sobie uświadomić, że firewalli nie rozwiążą wszystkich naszych problemów. Najslabszym ogniwem pozostaje zawsze człowiek, który swoim nawet przypadkowym działaniem może spowodować zniszczenie systemu. Nie ma znaczenia czy posiadamy firewalla czy też nie, ponieważ i tak nie uchroni nas to przed atakami. Firewall może je jedynie ograniczać kontrolując a zarazem zwiększając w jakimś tam stopniu bezpieczeństwo naszej sieci.”⁹

⁸ Internet Firewalls wyd. RM str.24

⁹ Internet Firewalls wyd. RM str.24

4. Usługi internetowe.

W tym rozdziale zrobimy krótki przegląd usług, z którymi możemy się spotkać podczas konfigurowania firewalla. Czasami słyszymy, że dana usługa jest zabezpieczona lub bezpieczna. Oznacza to, że dają nam one dwa rodzaje gwarancji:

- „Nie można użyć usługi w sposób niezgodny z jej przeznaczeniem lub też;
- Nikt nie może podejrzeć lub sfałszować transakcji.
- Można też używać „niezabezpieczonych” usług w bezpieczny sposób – trzeba po prostu zachować więcej ostrożności np. poczta elektroniczna za pośrednictwem SMTP. To klasyczny przykład usługi „niebezpiecznej”. Jeśli jednak odpowiednio skonfigurujemy serwery pocztowe i szyfrujemy wiadomości możemy osiągnąć zmianę poziomu bezpieczeństwa.”¹⁰

Krótki przegląd usług, z którymi można spotkać się podczas konfiguracji firewalla.

5. WWW

Popularność WWW zawdzięczamy przede wszystkim HTTP, jednakże wiele witryn używa różnych wtyczek, protokołów, które mają wpływ na bezpieczeństwo. Wielu użytkowników nie orientuje się w funkcjach i pochodzeniu sieci WWW, Netscape’a, Microsoft Internet Explorera, HTTP i HTML –u wszystkie te pojęcia są bardzo skomplikowane, co powoduje, że „narzędzia” te są łatwe w obsłudze dla użytkownika, ale bardzo skomplikowana budowa powoduje, że nie zawsze znamy wszystkie mechanizmy kierujące WWW.

Zagadnienia bezpieczeństwa klienta WWW

Przeglądarki WWW są bardzo popularne, ponieważ zapewniają intuicyjny interfejs graficzny i udostępniają olbrzymie zasoby Internetu. Niestety, trudno jest zabezpieczyć przeglądarki i serwery WWW. „Użyteczność WWW opiera się głównie

¹⁰ Internet Firewalls wyd. RM str 30

na prostocie obsługi i elastyczności. Jednakże te właśnie zalety powodują największe problemy w kontrolowaniu tej usługi. Łatwiej pobrać plik przy pomocy przeglądarki WWW niż używając do tego dosowego FTP. Dodatkowo przeglądarki obsługują technologie wtyczek (plug-ins), które rozszerzają możliwości przeglądarki. Jeśli do tego dodamy technologię Active X czy JavaScripts w tym momencie zdamy sobie sprawę jak z złożoną i trudną usługą mamy do czynienia”¹¹.

Zagadnienia Bezpieczeństwa serwera WWW

„Serwer WWW pozwala na wysyłanie poleceń każdej osobie, która może się z nim połączyć. Pliki te powinny być ogólnie dostępne, ale zarazem chronione. Jednakże wiele serwerów WWW oferuje wiele więcej funkcji niż wydawanie plików HTML. Wiele serwerów umożliwia konfigurację ich za pomocą przeglądarki WWW. Co może powodować duże problemy, jeżeli ty możesz konfigurować dany serwer ktoś inny też może to zrobić!”¹².

6. Poczta elektroniczna

„Jest to najbardziej popularna usługa sieciowa. Można by stwierdzić, że nie wnosi dużego ryzyka ale nie oznacza to, że jest zupełnie bezpieczna. Sfabrykowanie listu nie jest dużym problemem z reguły ma na celu:

- a) naruszenie reputacji
- b) manipulowanie ludźmi

Przyjmowanie poczty elektronicznej zabiera czas komputera i przestrzeń dyskową oraz naraża na atak blokowanie usług. Jeżeli dodamy do tego możliwości przesyłania pocztą plików to stwierdzimy, że poczta nie jest aż tak dobrze zabezpieczoną usługą. Jednakże bezpieczeństwo poczty zależy przede wszystkim od użytkownika (przecież on decyduje czy otworzyć daną przesyłkę lub załącznik). Serwery pocztowe najczęściej

¹¹ Internet Firewalls wyd. RM str 32

¹² Internet Firewalls wyd. RM str 34

używamy zależnie od danego systemu. W systemach uniksowych jest to Sendmail a w Windows NT/2000 - Microsoft Exchange. Obydwa te produkty nie są zbyt bezpieczne, o czym możemy się przekonać w Internecie gdzie, co chwilę odkrywa się nowe luki. Jednak nie ma systemów 100% bezpiecznych, musimy tylko zastosować odpowiednią politykę, która w wypadku złamania naszych zabezpieczeń ograniczyłaby szkody do minimum. Podczas gdy serwery do wymiany informacji używają protokołu SMTP to klienci, aby przesłać pocztę na serwer używają z reguły dwóch protokołów są to POP i IMAP. Problemem jest fakt, że dane autoryzacyjne są przesyłane bez żadnych zabezpieczeń (szyfrowanie). Co prawda istnieje możliwość zmuszenia tych protokołów do ukrywania informacji autoryzacyjnych ale ochrona samej zawartości listu pozostawia wiele do życzenia¹³.

7. Grupy dyskusyjne

Grupy dyskusyjne w Internecie są tak zaprojektowane, aby umożliwić komunikację „wielu do wielu”. Główny problem to przesyłanie informacji do użytkownika (możemy go porównać do komunikatu broadcastu), który nie powinien ich w ogóle otrzymać np. (hacker, który przedstawi nie swoją tożsamość i poprosi o podanie hasła logowania lub jego zmiany).

8. Transfer plików.

„Podstawowym protokołem do transmisji plików jest FTP (*File Transfer Protocol*). Większość przeglądarek ma już wbudowanego klienta FTP co powoduje, iż nawet nie obeznany użytkownik może pobierać pliki, np. z Internetu czy serwerów FTP. Pobieranie plików niesie z sobą duże ryzyko (konie trojańskie, wirusy). Jednak nie jest możliwe całkowicie pozbawienie danej sieci serwera FTP gdyż pełni on ważną rolę - może przechowywać pliki, które są potrzebne użytkownikom (artykuły, programy, drivery itp.). Najczęściej, aby umożliwić takie działanie stosuje się anonimowe FTP. Jest to serwer, który umożliwia nam pobieranie z niego plików. Ale jego zabezpieczenia pozostawiają wiele do życzenia. Autoryzacja polega na podaniu jako hasła adresu

¹³ Internet Firewalls wyd. RM str 35 - 36

e-maila, który nie zawsze musi być prawdziwy. Dodatkowo serwery takie mogą być wykorzystywane do przechowywania np. pirackiego oprogramowania itp. Dlatego należy dobrze rozpatrzyć umiejscowienie takiego serwera w sieci”¹⁴.

9. Współdzielenie plików

Współdzielenie plików pozwala na używanie plików znajdujących się na dyskach innych komputerów. Istnieje wiele protokołów obsługujących współdzielenie plików jednak muszą się one charakteryzować się pewnymi cechami muszą być przejrzyste. Najczęściej używane protokoły to: *Network File System* (NFS), w Uniksie lub Linuksie, *Common Internet File System* (CIFS) w Microsoft Windows i AppleShare w Macintoshach. CIFS stanowi część rodziny powiązanych protokołów i ma skomplikowane pochodzenie między innymi od *Server Message Block* (SMB), NetBIOS/NetBEUI i LanManager. Rozpatrując te wszystkie protokoły pod kątem widzenia firewalli wszystkie te protokoły są niezabezpieczone i trudne do używania w Internecie.

„NFS – został zaprojektowany do używania w sieciach LAN zakłada on szybkie odpowiedzi, dużą niezawodność, synchronizację czasu duży stopień zaufania pomiędzy maszynami. NFS ma wiele problemów z bezpieczeństwem niepoprawna konfiguracja może spowodować możliwość podmontowania systemu plików. Spowodowane jest to między innymi, tym że NFS używa uwierzytelniania na podstawie hostów, które to łatwo oszukać.

CIFS i AppleShare – posiadają lepsze uwierzytelnianie użytkowników, a nie jak w przypadku NFS na podstawie hostów. Jednakże AppleShare używa haseł wielokrotnych, co znacznie obniża poziom bezpieczeństwa. Nowe wersje CIFS stosują dobre uwierzytelnianie i zabezpieczanie. Jednak zgodność z poprzednimi wersjami powoduje podatność na ataki.”¹⁵

¹⁴ Internet Firewalls wyd. RM str 38 - 39

¹⁵ Internet Firewalls wyd. RM str 41

10. Drukowanie

„Prawie każdy dzisiejszy system umożliwia drukowanie zdalne poprzez *lp* i *lpr* w przypadku Uniksa lub Linuksa oraz SMB w Windows czy usługi drukowania AppleTalk w Macintoshach. Zdalne drukowanie umożliwia nam wysłanie danych do drukarki. Jednak raczej nie należy udostępniać możliwości drukowania poza siecią LAN. Ponadto istnieją błędy w niektórych systemach, co może powodować możliwość naruszenie bezpieczeństwa systemu”¹⁶.

(IRIX 6.2 – konto drukarki wierszowej jest puste)

11. Zdalny dostęp

Umożliwia korzystanie z zasobów innego komputera, których nie posiadamy na swoim komputerze. Najbardziej znanym terminalem jest Telnet jednak jest to usługa, której żaden administrator raczej nie powinien używać. Jednym z powodów jest przesyłanie haseł otwartym tekstem. Aby zwiększyć bezpieczeństwo należy używać szyfrowania sesji, najczęściej znane to SSH (*secure shell*) oraz VPN (*Virtual Private Network*). Oprócz dwóch wcześniej wymienionych używa się także *rlogin* i *rsh*, jednak odkryte błędy nie zalecają używania wyżej wymienionych.

12. Zdalne interfejsy graficzne

Z pośród rodziny produktów dla platformy Windows Microsoft dostarcza zdalny interfejs graficzny oparty na protokole Remote *Desktop Protocol* (RDP). Z bardziej znanych są to LapLink pcANYWHERE. Dla maszyn uniksowych, czyli dla Linuksa dostępny jest system graficzny X11. Serwery X11 są kuszącym celem, na jednej maszynie istnieje możliwość uruchomienia 64 takich serwerów, dostęp jest możliwy poprzez modyfikacje *xhost* dopisaniu się do tego pliku. Można również przechwycić polecenia z portów 6000:6065 i przejąć kontrolę nad klawiaturą i myszką .

¹⁶ Internet Firewalls wyd. RM str 42

13. Usługi konferencyjne czasu rzeczywistego

Z pośród najbardziej znanych usług tego typu jest *Internet Relay Chat* (IRC). Z reguły komunikacja odbywa się za pomocą specjalnych klientów IRC lub też Telnetu. IRC powoduje wiele problemów z bezpieczeństwem z reguły ma to związek z klientami IRC-a. Wielu klientów pozwala serwerowi na zbyt duży dostęp do lokalnych zasobów dodatkowo większość najaktywniejszych użytkowników IRC-a to krakerzy. Osobną rolę spełniają systemy wideo konferencyjne, najbardziej znane to NetMeeting protokoły te są trudne do zabezpieczenia z powodu używania wielu strumieni danych a także braku wyraźnego podziału klient –serwer. Dodatkowo używają protokołów bezpołączeniowych i portów przyznawanych dynamicznie w celu wymiany danych, które są bardzo trudne do filtrowania. Jeżeli musimy je posiadać w sieci to wyznaczmy osobny serwer dla takich usług poza naszą siecią wewnętrzną.

14. Usługi nazewnicze i katalogowe

Z pośród najbardziej znanych to *Domain Name System* (DNS) oraz znane ze środowiska firmy Microsoft *DNS + Active Directory* (Win2000) a dla starszych sieci *Windows Intranet Name Service* (WINS).

DNS – podstawowy protokół używany w Internecie umożliwia zamianę adresów IP na nazwę hostów i odwrotnie. Główne ryzyko przy używaniu DNS to dostarczenie informacji nie tam gdzie chcemy ją dostarczyć. Poza tym DNS pozwala dołączać informacje o używanym sprzęcie i oprogramowaniu. Sieci oparte na Windows 2000 do lokalizacji zasobów używają DNS w połączeniu z usługą Active Directory poprzez protokół *Lightweight Directory Access Protocol* (LDAP).

WINS – są używane do translacji nazw hostów NetBIOS na adresy IP. Wadą tego rozwiązania jest niemożliwość funkcjonowania poza sieciami LAN. Należy tu zauważyć, że WINS ma jeszcze większe problemy z bezpieczeństwem niż DNS, ponieważ przekazuje nazwy użytkowników i listy uruchomionych usług. WINS opiera się na dynamicznej rejestracji.

Wcześniejsze wersje BINDA nie były odporne na włamania, istniała możliwość nieautoryzowanego transferu strefy. Słabe zabezpieczenia przy uwierzytelnianiu innych serwerów DNS spowodowały dopracowanie tego oprogramowania. Od wersji BIND 9 sytuacja się zmieniła jednak problem polega na tym że wszystkie te dane można podsłuchać najlepszym rozwiązaniem była by implementacja na serwerach DNS SSL, TSL lub VPN. Oczywiście można dzięki konfiguracji DNS-a ustawić prawa transferu strefy dla wybranych serwerów ale napastnik zawsze może podszyć się pod danego hosta.

15. Strategie zabezpieczeń

Zanim przystąpimy do konfiguracji firewalla musimy przeanalizować strategię zabezpieczeń, która umożliwi nam odpowiednie skonstruowanie naszego firewalla.

1. Minimalne przywileje (least privilege)

Strategia ta polega na przyznaniu takich przywilejów jakie dany użytkownik potrzebuje i są mu niezbędne do wykonywania zadań – i żadnych więcej nie powinien posiadać. Jako przykład możemy posłużyć się Sendmailem, który działa przez część czasu z prawami roota.

2. Dogłębna obrona (defense in depth)

Zasada jest w tym wypadku krótka.

Nie należy polegać tylko na jednym zabezpieczeniu !.

W tym przypadku mechanizmy zabezpieczeń nawzajem się wspomagają. Można to osiągnąć przez zabezpieczenie sieci (firewall), zabezpieczanie hostów (zwłaszcza hostów bastionowych) bezpieczeństwo związane z ludźmi (edukacja użytkowników, ostrożna administracja systemem itd.). Sprawdza się tu powiedzenie, że system bezpieczeństwa jest tak bezpieczny jak najsłabsze ogniwo w tym systemie.

3. Wąskie przejście (*choke point*)

Zmusza on napastników do używania kanałów, które w łatwy sposób można kontrolować i monitorować. Takie zadanie spełnia między innymi firewall jednak wąskie przejście staje się bezużyteczne jeżeli posiadamy wejście „kuchennymi drzwiami”.

Jako punktę czym powinniśmy się kierować przy konstruowaniu strategii zabezpieczeń niech będzie:

CO NIE JEST DOZWOLONE JEST ZABRONIONE

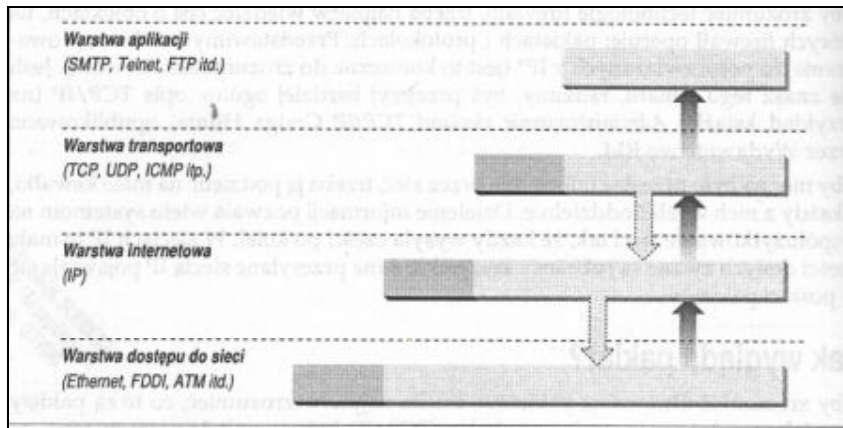
II. Budowa ścian ogniowych

„Aby zrozumieć technologię firewalli musimy najpierw zapoznać się z obiektami, na których działa firewall, są to między innymi pakiety i protokoły. Postaram się je tutaj pokrótce omówić gdyż bez zrozumienia tego nie jesteśmy w stanie zorientować się co tak naprawdę robi firewall.

Aby można było przesłać jakąkolwiek informację przez sieć należy podzielić ją na małe kawałki, które w sieciach *IP* nazywamy pakietami. Pakiety są konstruowane dla warstw OSI. Na każdym poziomie pakiet składa się z dwóch części: nagłówka i treści. Nagłówek zawiera informacje odnoszące się do warstwy, a treść zawiera dane dla tej warstwy, które składają się z całego pakietu następnej warstwy. Każda warstwa traktuje informacje, które dostała z warstwy znajdującej się wyżej, jako dane i dokłada do nich własny nagłówek. Na każdym poziomie pakiet zawiera informacje z poprzednich warstw, nic nie jest tracone. Proces zachowywania danych i dodawania własnego nagłówka jest nazywany *enkapsulacją*¹⁷.

¹⁷ Internet Firewalls wyd. RM str 69

Rys.1 Enkapsulacja danych



Źródło: *Internet Firewalls* wyd. RM str.70

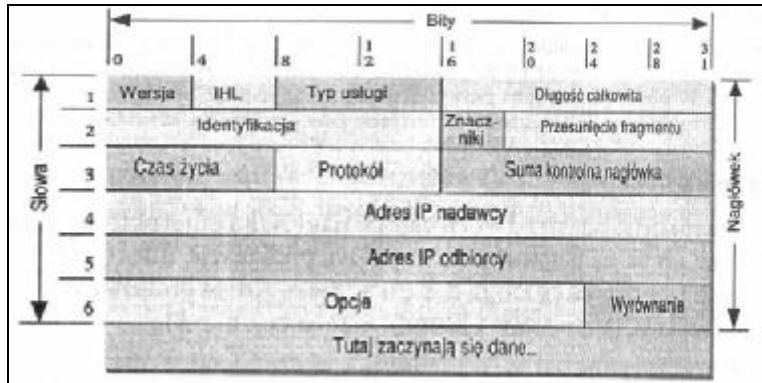
Warstwa Ethernetu składa się z dwóch części nagłówka ethernetowego i zawartości. Jako adres ethernetowy używamy adresu *MAC*.

Nagłówek z reguły określa:

- Rodzaj pakietu - np. *IP* ale może to być również AppleTalk, Novell
- Adres ethernetowy maszyny (nadawcy) jeżeli nie jest to w tej samej sieci adresem będzie ruter, który nadał informacje.
- Adres ethernetowy maszyny, do którego ma być wysłany pakiet. Istnieje możliwość, że będzie to informacja nadana w trybie *broadcastu* lub *multicastu*.

Warstwa IP – pakiet składa się z dwóch warstw: nagłówka *IP* i zawartości.

Rys.2 Nagłówek *IP* i zawartość pakietu



Źródło: *Internet Firewalls* wyd. RM str.72

Omówienie całego wyglądu pakietu *IP* nie jest tutaj nam potrzebne dlatego skupmy się na opcjach ważnych z punktu widzenia budowy firewalli.

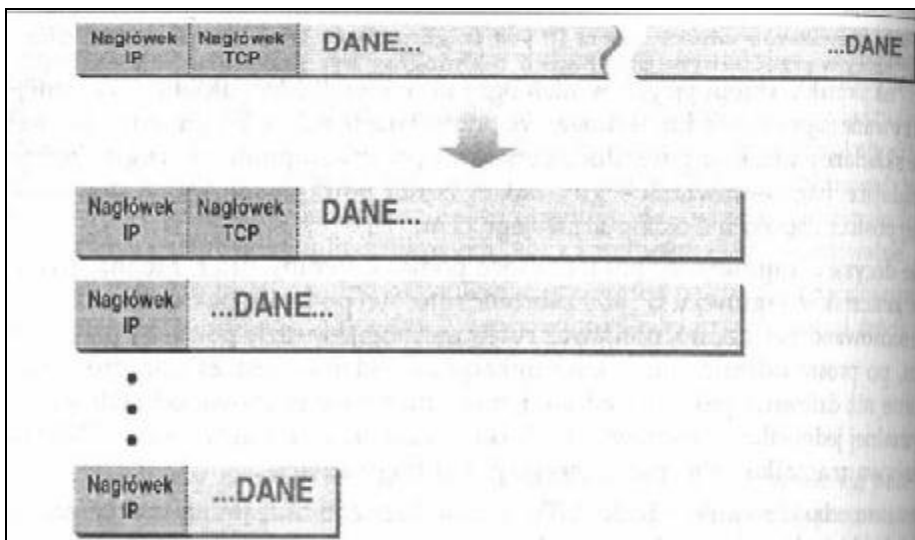
Opcje *IP*

Nagłówki *IP* mogą mieć ustawione pewne opcje. W założeniu miały one pomagać w przesyłaniu pakietów przez sieć. Jednak opcje te umożliwiają obejście pewnych zabezpieczeń. Najczęściej spotykaną opcją jest opcja trasowania źródłowego (*source routing*). Pozwala ona na określenie dokładnej trasy jaką ma przebyć pakiet od nadawcy do adresata.

Fragmentacja IP

IP może podzielić duży pakiet na mniejsze w celu przesłania przez sieć, która nie akceptuje danej wielkości pakietu. Fragmenty te są następnie składane przez komputer docelowy. Zwykle decyzje o fragmentacji podejmuje ruter

Rys.3 Fragmentacja danych.



Źródło: Internet Firewalls wyd. RM str.76

„Z punktu widzenia filtrowania problem polega na tym, że tylko pierwszy fragment zawiera informacje z protokołu wysokiego poziomu (*TCP*), które system filtrowania wykorzystuje do podjęcia decyzji, czy pozwolić na przepuszczenie całego pakietu. Początkowo systemy filtrowania pakietów filtrowały tylko pierwszy jego fragment jako fragment, a pozwalały na przedostawanie się pozostałym. Jednak nie jest to bezpieczne gdyż pofragmentowane dane obecne są w pamięci danego hosta, który czeka na pozostałe pakiety co może spowodować blokadę usług. Gdy host docelowy zrezygnuje ze składania pakietu wyśle informacje ICMP „przekroczono czas składania pakietów” dzięki temu napastnik zorientuje się, że host istnieje. Ponadto istnieje możliwość podzielenia pakietów w celu ukrycia przesyłanych danych. Dodatkowo

można tak skonstruować pakiet aby dane fragmenty pakietów pokrywały się. Systemy operacyjne bardzo różnie reagują na takie dane, z reguły zawieszają się co umożliwia następujące rodzaje ataków¹⁸:

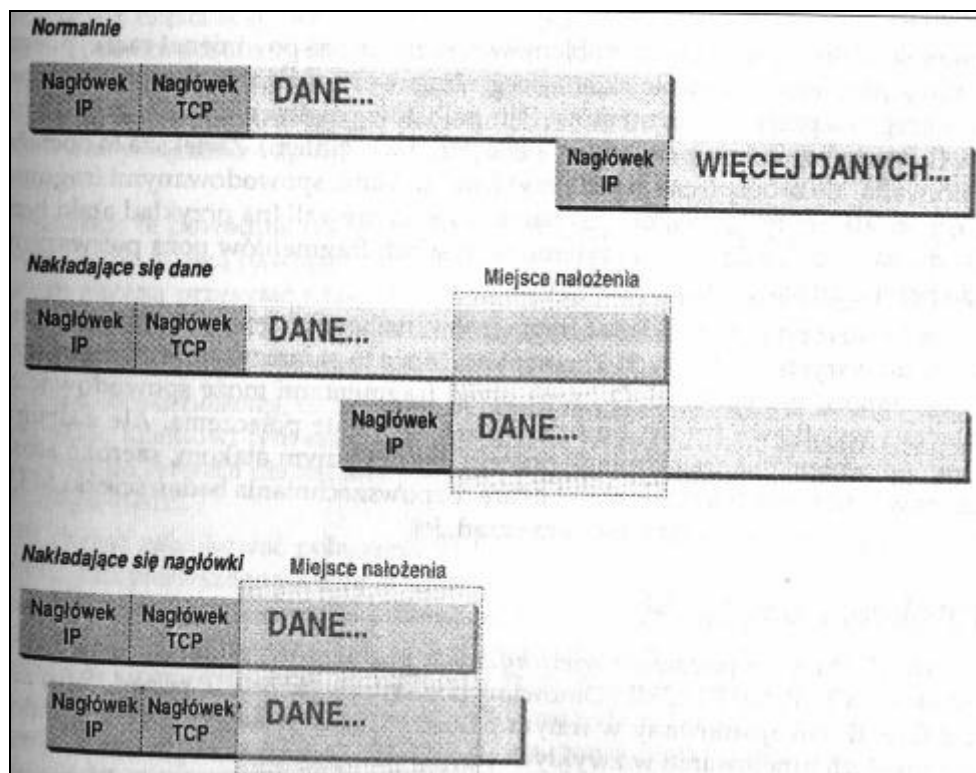
Blokada usług

Ataki ukrywające informacje (ustawienie aby pakiet został składany od numeru 5) niektóre programy antywirusowe sprawdzają przesyłane informacje tylko w pierwszy pakiecie (0) traktując pozostałe jako dobre.

Ataki przesyłające informacje do zablokowanych portów. Napastnik może utworzyć pakiet z akceptowalnym nagłówkiem w pierwszym fragmencie a następnie nałożyć na niego drugi fragment.

¹⁸ Internet Firewalls wyd. RM str 79 - 76

Rys.4 Nakładające się fragmenty



Źródło: *Internet Firewalls* wyd. RM str.77

Protokół TCP

Jest to najczęściej wykorzystywany protokół w Internecie (Telnet, FTP, SMTP, NNTP itp.). *TCP* jest niezawodny ponieważ :

- Cel odbierze dane aplikacji w takiej kolejności, w jakiej były wysłane.
- Cel odbierze wszystkie dane aplikacji.
- Cel nie otrzyma żadnych duplikatów informacji .

Opcje TCP

W przesyłaniu protokołem *TCP* stosuje się kilka rodzajów opcji są to:

URG (*urgent* – pilne)

ACK (*acknowledgment* – potwierdzenie)

PSH (*push* – przekaż)

RST (*reset* –zresetuj)

SYN (*synchronize* – synchronizacja)

FIN (*finish* – zakończ)

Z tych wszystkich pozycji najbardziej interesujące są dla nas *ACK* i *RST* (*ACK*, ponieważ umożliwia zdefiniowanie pierwszego pakietu połączenia. Zaś *RST* ponieważ jest dobrym sposobem odmawiania połączeń bez zwracania błędu). Jednak wiele implementacji *TCP/IP* nieprawidłowo reaguje na dziwne kombinacje przy zmianach tych opcji np. zawieszenie maszyny . Ponadto znając implementacje *TCP/IP* narzędzia takie jak *nmap*, *nessus* mogą odgadnąć jakiego systemu operacyjnego używamy na podstawie skanowania *fingerprint*.

ICMP

Używany do komunikatów kontrolnych i informacyjnych *IP* oraz diagnozowania problemów z siecią. Pakiety *ICMP* są przesyłane w *IP*.

Przykładowe komunikaty *ICMP* to:

Żądanie echa (*echo request*) – ping

Odpowiedź echa (*echo response*) – odpowiedz na ping

Przekroczenie echa (*time exceeded*) – liczba ruterów, przez które przeszedł pakiet.

Nieosiągalne miejsce przeznaczenia (*destination unreachable*)

Przekierowanie (*redirect*)

Najbardziej znane wykorzystanie poza diagnostyką sieci to atak odmowy usług tzw. *ping of death* .

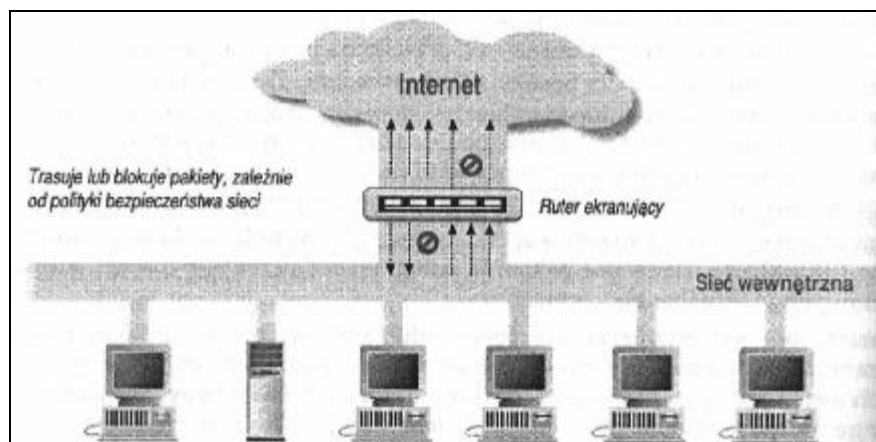
III. Architektura firewalli

Najprostsza architektura to pojedynczy obiekt (*single-box architecture*) działający jako firewall. Zaletą tego rozwiązania jest to, że mamy wszystko w jednym. Rozwiązanie takie ma jednak wady są to: bezpieczeństwo takiej sieci zależy od jednego urządzenia co powoduje, że jest to najsłabsze ogniwo. W praktyce zaletą pojedynczej architektury są względy praktyczne nie musimy konfigurować kilku maszyn tylko jedną. Jednak w przypadku udanego ataku napastnik ma dostęp do całej sieci bez żadnych ograniczeń.

3.1 Ruter osłaniający

Użycie samego rutera (*screening router*) jako filtra pakietów, który chroni całą sieć jest rozwiązaniem tanim. Nie jest to rozwiązanie zbyt elastyczne gdyż można tylko akceptować lub odrzucać protokoły na podstawie numerów portów, ale trudno zabraniać pewnych operacji w ramach danego protokołu. Dodatkowo brak drugiej linii obrony powoduje to, że jeżeli ruter zostanie złamany nie ma dalszych zabezpieczeń.

Rys.5 Używanie rutera ekranującego do filtrowania pakietów



Źródło: Internet Firewalls wyd. RM str.98

Ruter ekranujący nadaje się na firewall w następujących sytuacjach :

- Chroniona sieć ma dobrze zabezpieczone hosty
- Liczba używanych protokołów jest ograniczona
- Maksymalna wydajność i nadmiarowość

Rozwiązania takie są użyteczne w firewallach internetowych i sieciach, które dostarczają usług internetowych.

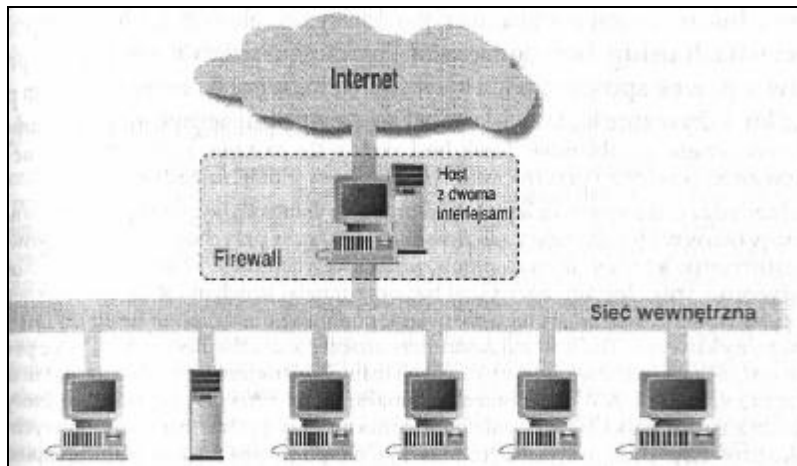
3.2 Host dwusieciowy

Rozwiązanie takie polega zbudowaniu hosta z dwoma interfejsami sieciowymi (*dual-homed host*). Może on służyć jako ruter pomiędzy sieciami, który ma dwa interfejsy. Ma możliwości trasowania pakietów *IP* z jednej sieci do drugiej. Jednak aby używać takiego hosta jako firewalla, trzeba włączyć funkcję trasowania . Tak więc pakiety *IP* z jednej sieci przesyłane są do drugiej. Systemy, które są chronione mogą komunikować się z firewallem i z hostem, który jest za ścianą ogniową. Jednak systemy te nie mogą bezpośrednio komunikować się ze sobą (ruch *IP* pomiędzy nimi jest zablokowany), niektóre odmiany z tej architektury używają *IP* w Internecie, a w sieci wewnętrznej jakiegoś innego protokołu (*NetBEUI, IPX/SPX*). Zaletą tego rozwiązania jest bardzo wysoki stopień kontroli(nie możemy dokonać bezpośredniego routingu), do wad zaliczymy małą wydajność. Musimy odpowiednio zabezpieczyć się, gdyż opanowanie takiego hosta przez napastnika powoduje to, że ma on dostęp do całej sieci.

Host z dwoma interfejsami nadaje się na firewall w następujących sytuacjach :

- Ruch do Internetu jest niewielki
- Łączność z Internetem nie decyduje o prowadzeniu firmy
- Nie udostępniamy użytkownikom Internetu żadnych usług
- Chroniona sieć nie zawiera wartościowych danych

Rys.6 Architektura hosta dwusieciowego



Źródło: *Internet Firewalls* wyd. RM str.117

3.3 Rozwiązania wielofunkcyjne

Można je scharakteryzować „wszystko w jednym”. Oferują pewną kombinację pośredniczenia i filtrowania pakietów a zarazem łączy zalety obydwu rozwiązań. Możemy zezwolić na wydajną obsługę części protokołów, a inne dokładnie kontrolować.

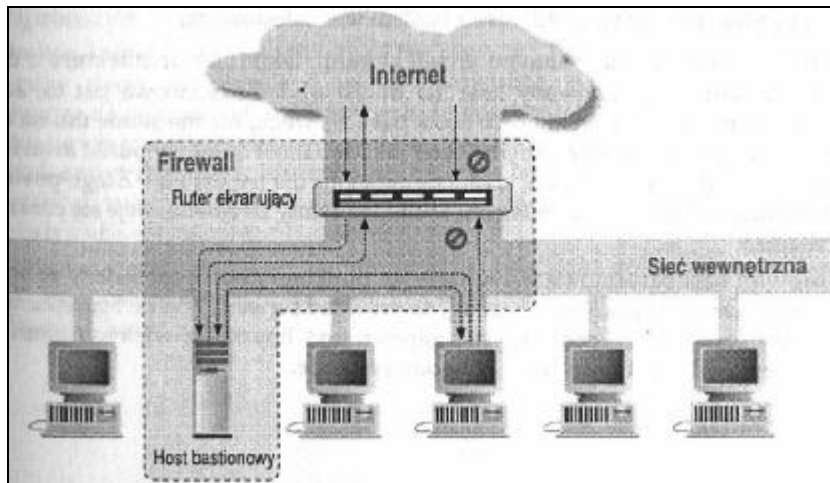
Takie rozwiązanie odpowiednie jest w następujących sytuacjach :

- Sieć jest mała
- Nie udostępniamy żadnych usług w Internecie

3.4 Architektura ekranowanego hosta

Polega na udostępnieniu hosta, który przyłączony jest do sieci wewnętrznej za pomocą oddzielnego rutera. Rozwiązanie takie bazuje na filtrowaniu pakietów (filtrowanie pakietów zabezpiecza przed obchodzeniem serwerów proxy i tworzeniu bezpośrednich połączeń).

Rys.7 Architektura z ekranowanym hostem



Źródło: *Internet Firewalls* wyd. RM str.119

Host bastionowy jest umiejscowiony w sieci wewnętrznej. Filtrowanie pakietów w ruterze ekranującym skonfigurowano tak, by host bastionowy był jedynym hostem w sieci wewnętrznej, z którym mogą łączyć się hosty z Internetu. Każdy zewnętrzny system próbujący dostać się do wewnętrznych systemów lub usług musi połączyć się z tym hostem. Dzięki filtrowaniu pakietów host bastionowy może nawiązać dozwolone połączenie ze światem. Ruter ekranujący może być tak skonfigurowany aby wykonywał jedno z poniższych działań :

- Pozwalał hostom wewnętrznym na nawiązanie połączeń z hostami w Internecie na potrzeby pewnych usług
- Zabraniał nawiązania połączenia z wewnętrznymi hostami

Wadą tego rozwiązania jest fakt, iż w wypadku włamania do hosta bastionowego nic nie stanie na drodze w penetracji hostów wewnętrznych. Dodatkowo w wypadku awarii, któregoś z urządzeń (ruter lub host) cała sieć ulega awarii. Co uniemożliwi nam połączenie z niektórymi usługami w Internecie.

Architekturę taką możemy zastosować w następujących sytuacjach :

- Gdy niektóre połączenia nadchodzą z Internetu.
- Gdy ochraniać sieć ma dobrze zabezpieczone hosty.

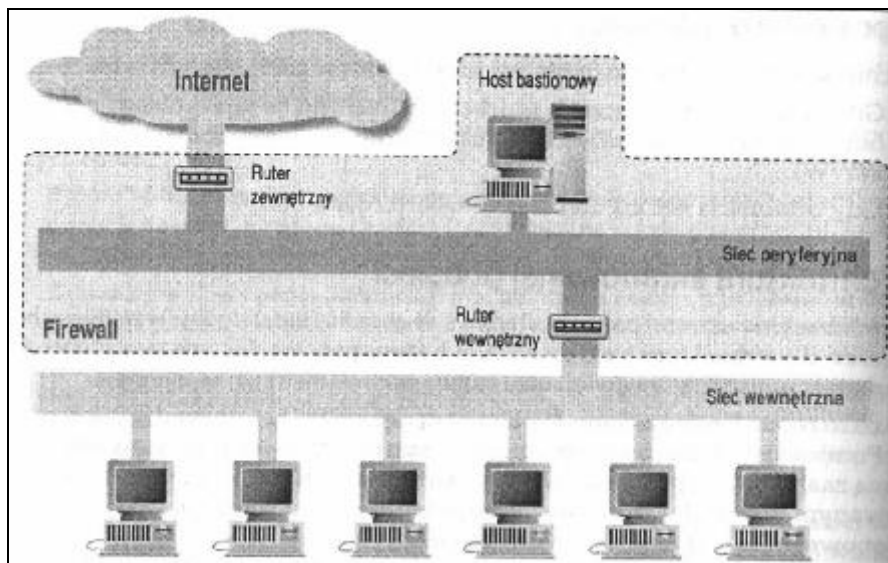
3.5 Architektura ekranowanej podsieci

Jest to dodanie dodatkowej warstwy zabezpieczeń do architektury z ekranowanym hostem poprzez dodanie sieci peryferyjnej (*perimeter network*).

Możemy zadać sobie pytanie dlaczego wykonujemy taką operację ?.

Ponieważ hosty bastionowe są z natury najbardziej narażonymi hostami w sieci. Dodanie dwóch ruterów ekranujących pomiędzy siecią peryferyjną a siecią wewnętrzną, zaś drugiego między siecią peryferyjną a siecią zewnętrzną spowoduje, że w wypadku włamania napastnik musi przejść przez dwa routery. Dodatkowo w wypadku włamania do hosta bastionowego zawsze pozostaje drugi router wewnętrzny.

Rys.8 Architektura ekranowanej podsieci (używająca dwóch podsieci).



Źródło: *Internet Firewalls* wyd. RM str.122

Sieć peryferyjna

Sieć peryferyjna (*perimeter network*) pełni rolę następnego zabezpieczenia oddzielając sieć zewnętrzną od sieci wewnętrznej, dodatkowo zabezpiecza to sieć przed podsłuchaniem. W tym wypadku napastnik może ewentualnie podsłuchać ruch, który

odbywa się w sieci peryferyjnej. Ponieważ żaden dodatkowy ruch nie przechodzi przez sieć peryferyjną z sieci wewnętrznej.

Host bastionowy

Dołączenie hosta bastionowego do sieci peryferyjnej ma na celu ustalenie głównego kontaktu połączeń przechodzących z zewnątrz np:

- Sesja przychodzącej poczty (*SMTP*)
- połączeń przychodzących przez *FTP* do serwera anonimowego
- przychodzących zapytań *DNS*

Usługi wychodzące obsługiwane są na zasadzie:

- Ustawienie filtrów pakietów na zewnętrznym i wewnętrznym routerze, pozwalające wewnętrznym klientom na bezpośredni dostęp do zewnętrznych serwerów.
- Ustawienie serwerów proxy w hoście bastionowym

Filtrowanie pakietów pozwala hostowi bastionowemu na połączenie się z Internetem oraz akceptowanie połączeń hostów z Internetu.

Ruter wewnętrzny (*interior router*) nazywany też ruterem dławiącym (*choke router*), zabezpiecza sieć wewnętrzną zarówno od strony Internetu jak i od sieci peryferyjnej. Ruter wewnętrzny filtruje większość pakietów do firewalla. Pozwala to na wydzielenie usług wychodzących z sieci wewnętrznej do Internetu. Są to usługi z reguły bezpieczne, które nie spowodują osłabienia zabezpieczenia sieci np. wychodzące połączenia *HTTP*, *SMTP*, *POP3* itp. Jednak nie można pozwalać, aby znalazły się tam usługi, których nie potrzebujemy. Jeżeli nie używamy *FTP* to nie udostępniamy tej usługi.

Ruter zewnętrzny (*exterior router*) zwany też ruterem dostępowym ma za zadanie ochronę sieci peryferyjnej i sieci wewnętrznej. Z reguły routery te pozwalają wszystkim danym wychodzącym wydostać się na zewnątrz prawie nie filtrując pakietów dlatego, iż reguły pakietów chroniące maszyny wewnętrzne musiałyby być takie same jak dla drugiego routera. W takim wypadku błąd w regułach filtrowania na jednym routerze spowodowałby możliwość obejścia zabezpieczeń drugiego routera.

Tak więc dlaczego i po co jest ten ruter?. Ma on za zadanie chronić maszyny w sieci peryferyjnej (*DMZ*) a także nie dopuścić do ruchu pomiędzy hostami wewnętrznymi a Internetem. Dodatkowo może on blokować pakiety z nieprawidłowym adresem źródłowym czy też fałszywym adresem źródłowym. Architektura ta jest uniwersalna i najczęściej stosowana ze względu na dużą funkcjonalność.

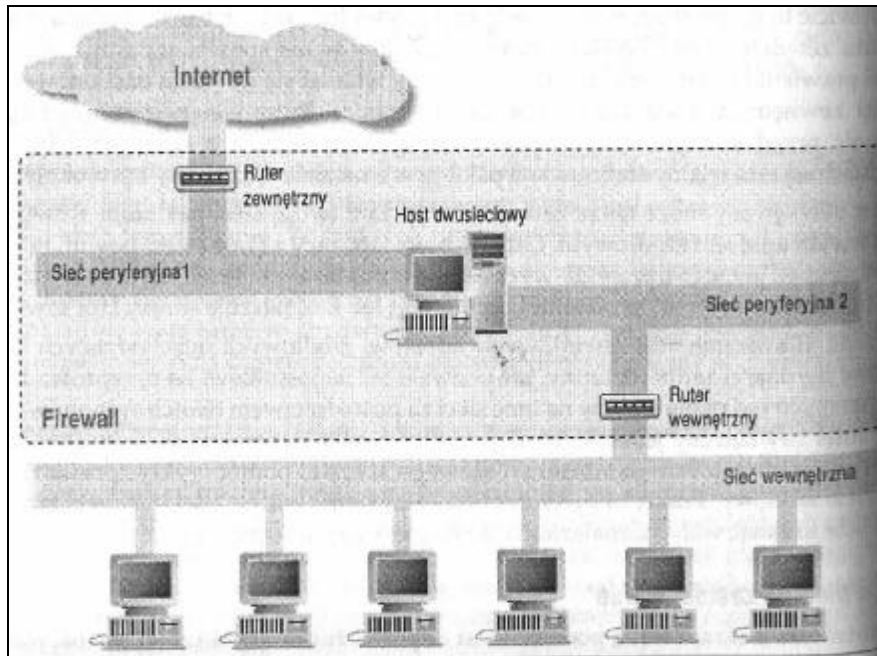
3.6 Architektury z wieloma ekranowanymi podsieciami

Rozwiązania w przypadku tej architektury będą opisane poniżej mają one zastosowania w zaawansowanych firewallach. Są modyfikacjami rozwiązań opisanych w wcześniejszych punktach.

3.7 Wieloczęściowa sieć ekranowana (split-screened subnet)

Jest podobna do poprzedniej konfiguracji, ale dodatkowo dodane są sieci z reguły połączone z pomocą jednego lub więcej hostów wielosieciowych zamiast ruterów. Czasami używa się tego rozwiązania w celu ochrony hostów z proxy za pomocą ruterów a także zapewnić wielowarstwowe zabezpieczenia. Rutery zapewniają ochronę przed fałszerstwami lub takimi jak awariami hosta. Jest to bardzo dobre rozwiązanie dające nam pełny komfort bezpieczeństwa, ale wymaga starannej konfiguracji hosta wielosieciowego.

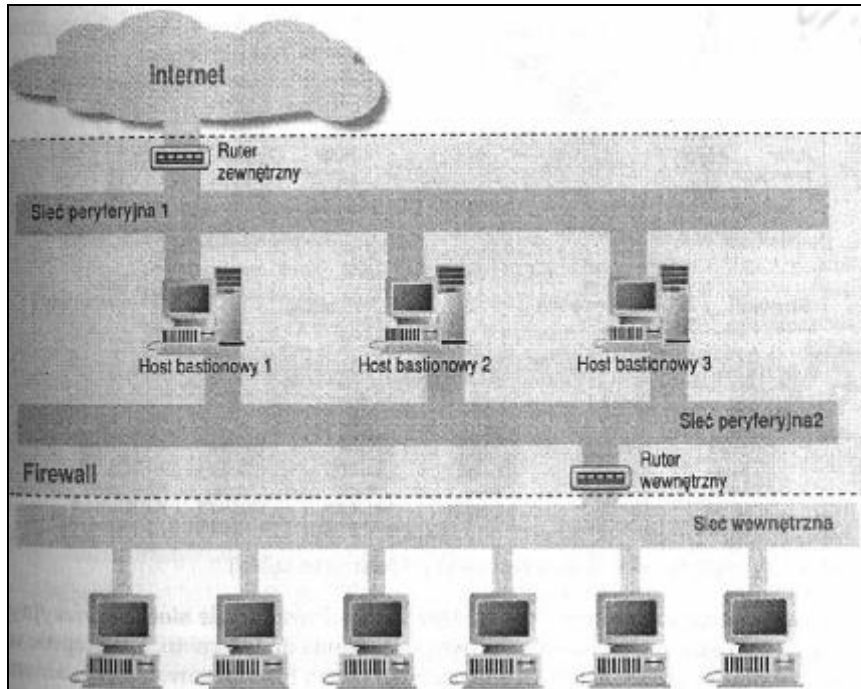
Rys. 9 Dwuczęściowa ekranowana sieć z hostem z wieloma przyłączeniami.



Źródło: *Internet Firewalls* wyd. RM str.126

Czasami stosuje się taką architekturę, żeby umożliwić administrowanie komputerami, które dostarczają usługi dla Internetu. Pozwala to na używanie przez administratorów protokołów, które nie są zbyt bezpieczne w Internecie.

Rys. 10 Dwuczęściowa ekranowana sieć bez przechodzącego ruchu.



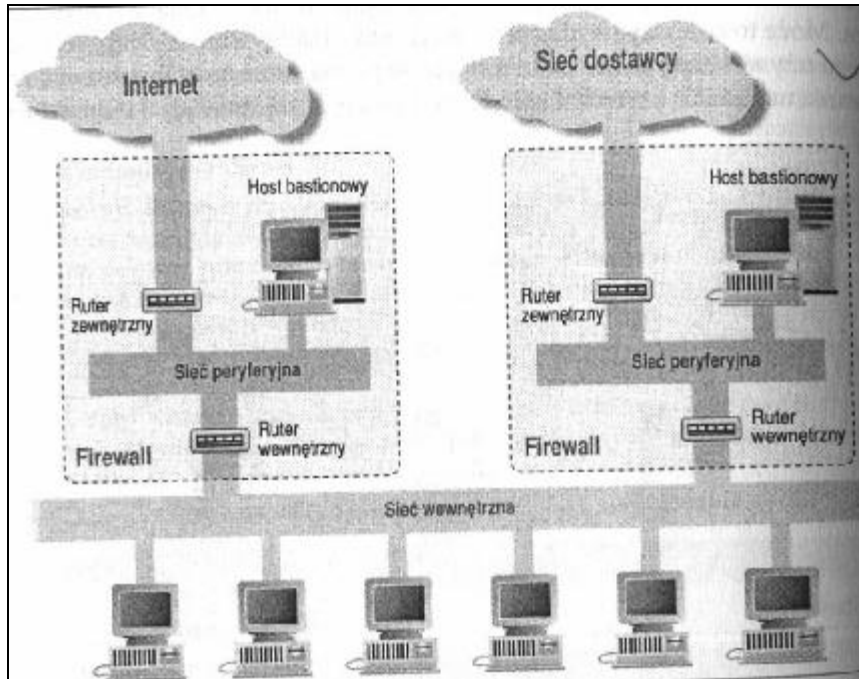
Źródło: *Internet Firewalls* wyd. RM str.127

Ten rodzaj zabezpieczeń (podsieci ekranowane) jest odpowiedni dla sieci wymagających wysokiego poziomu bezpieczeństwa zwłaszcza, jeśli dostarczają usług w Internecie.

3.8 Niezależne ekranowane podsieci (independent screened subnet)

Czasami musimy posiadać wiele niezależnych podsieci ekranowanych z oddzielnymi zewnętrznymi ruterami

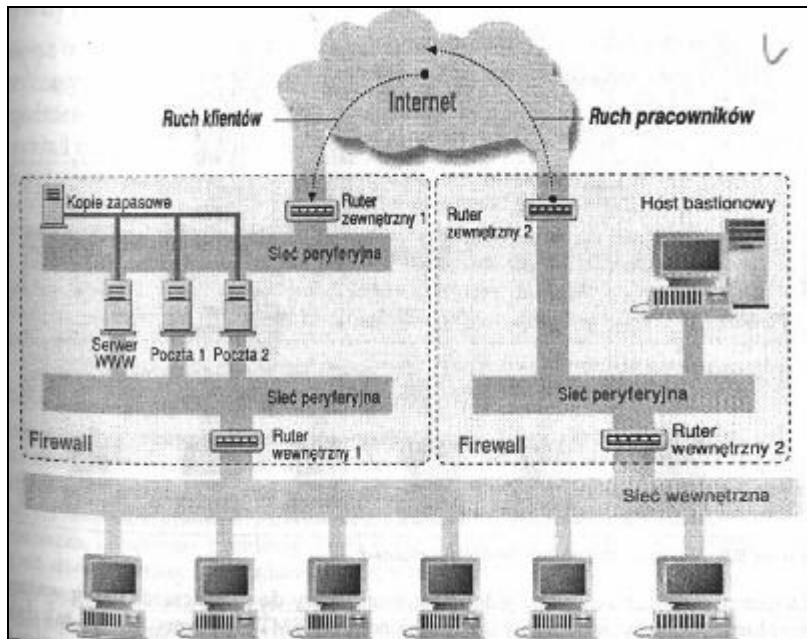
Rys. 11 Architektura z wieloma sieciami peryferyjnymi (wiele firewalli).



Źródło: *Internet Firewalls* wyd. RM str.128

Rozwiązanie takie można zastosować w wypadku, gdy mamy poufne dane, które przesyłamy siecią firmową. A pozostałe informacje przesyłamy przez Internet. Dodatkowo możemy rozdzielić dane segmenty sieci pod kątem usług, które udostępniamy (*WWW, FTP* itp.) Rozwiązanie takie ma tą zaletę, że każda usługa jest jakby w innej sieci jednocześnie będąc połączonym w jednolitą całość

Rys. 12 Przykład zaawansowanego firewalla.



Źródło: *Internet Firewalls* wyd. RM str.129

Rozwiązanie przedstawione na rys.12 ma zastosowanie w dużych sieciach korporacyjnych gdzie wymagany jest duży stopień bezpieczeństwa a także nadmiarowość i skalowalność.

IV. Ataki oparte na szczegółach niskopoziomowych protokołów

To tylko najbardziej znane ataki, jednakże sposób ich wykonania może nam pokazać, czym należy się kierować konfigurując firewall.

4.1 Skanowanie portów

Proces polegający na poszukiwaniu otwartych portów maszyny, aby zorientować się, które z nich można zaatakować. Proste skanowanie portów jest dość łatwe do wykrycia, jednak istnieje sposób zakamuflowania ich. Wiele maszyn nie rejestruje połączeń dopóki nie zostaną w pełni nawiązane, tak więc napastnik może wysłać pakiet początkowy z ustawionym bitem SYN, ale bez ACK a następnie odebrać odpowiedź z flagą SYN jeżeli port jest otwarty lub RST jeśli port jest zamknięty, jest to skanowanie półotwarte (*half open scan*). Pomimo braku rejestracji działanie takie może mieć efekty uboczne, zwłaszcza, jeżeli skaner nie wyśle na koniec pakietu z flagą RST. Istnieje również możliwość skanowania otwartych portów wysyłając pakiety na taki port, jeżeli dostaniemy odpowiedź z flagą RST dowiemy się o stanie danego portu (czy jest zamknięty lub otwarty). Prawie każda kombinacja flag poza SYN może być użyta do tego celu, chociaż najbardziej popularnymi opcjami są np: pakiety z flagą FIN. Czasem zwane odpowiednio *choinką* (*christmas tree*, ponieważ niektóre urządzenia sieciowe pokazują opcje za pomocą diod, a w tym wypadku zapalają się wszystkie jak na świątecznej choince). Mogą one spowodować efekty uboczne dla słabych implementacji stosów *TCP/IP*. Wiele urządzeń zawiesi się lub wyłączy *TCP/IP*. Dodatkowo istnieje możliwość na podstawie analizy stosu *TCP/IP* rozpoznanie rodzaju systemu operacyjnego (*questo*, *nmap*) co pozwala na szybsze dostosowania ataków zależnie od platformy.

4.2 Słabość implementacji

Wiele z ataków na tym poziomie to blokady usług, które wykorzystują słabości implementacji TCP/IP do zawieszenia maszyn. Na przykład *łza* (*teardrop*) i jej pochodne wysyłają nakładające się fragmenty. Istnieją także ataki, polegające na wysłaniu nieprawidłowych kombinacji opcji, ustawianiu nieprawidłowych długości pól czy oznaczaniu danych jako pilne tam, gdzie jest to niespodziewane (*niszczyciel Windows- winnuke*). Ustawiając odpowiednie flagi w nagłówkach *TCP/IP* możemy spowodować zawieszenie się danej maszyny (*hping*).

4.3 Podszywanie się

Podszywanie się (*spoofing*), polega na wysłaniu pakietów z nieprawidłowym adresem źródłowym (*hping*), co powoduje, iż wysłana informacja trafia do pozornego nadawcy a nie do atakującego. W podszywaniu się możemy wyróżnić trzy przypadki:

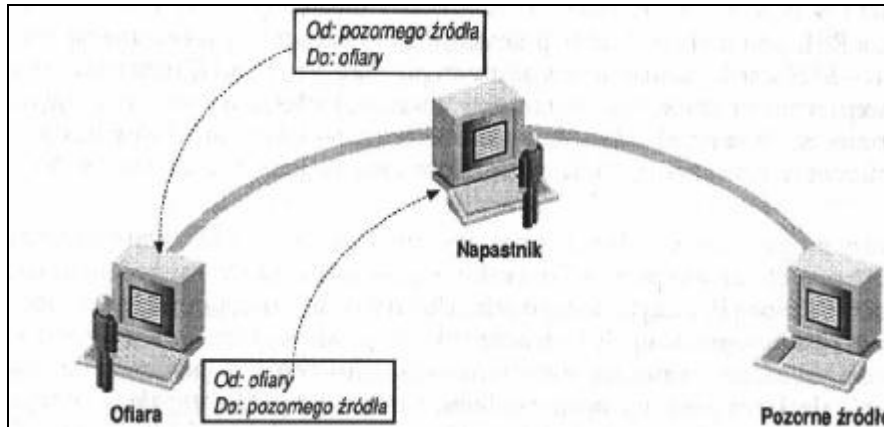
1. Napastnik może przejąć odpowiedź.
2. Napastnik nie potrzebuje odpowiedzi.
3. Napastnik nie chce odpowiedzi.

Celem ataku jest skierowanie odpowiedzi w inne miejsce.

Ad 1)

Jeśli napastnik znajduje się pomiędzy celem a pozornym źródłem to ma możliwość zobaczenia odpowiedzi i dalszego prowadzenia konwersacji. Sposób ten jest używany w ataku przejmowania połączeń.

Rys.13 Napastnik odbiera odpowiedź na sfabrykowane pakiety.

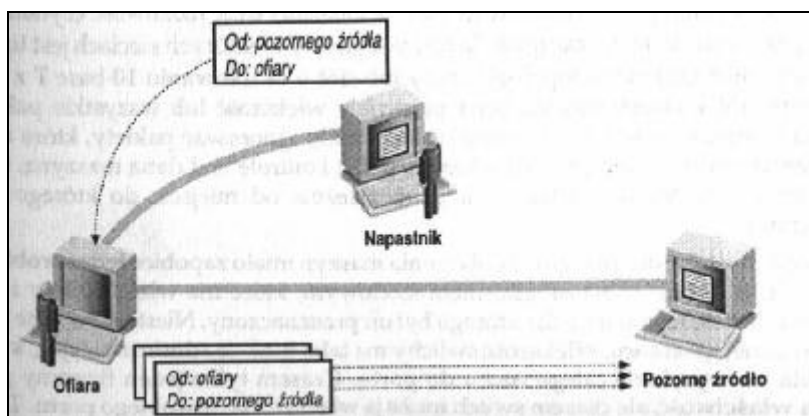


Źródło: *Internet Firewalls* wyd. RM str.90

Ad 2)

Napastnika nie zawsze interesuje odpowiedź. Jeżeli atak ma na celu zablokowanie usługi, host atakowany i tak nie jest w stanie odpowiedzieć.

Rys. 14 Napastnik używa sfabrykowanych pakietów do blokowania usług.



Źródło: *Internet Firewalls* wyd. RM str.91

Ad 3)

Cześć ataków opiera się na założeniu, że odpowiedź (a jeszcze lepiej mnóstwo odpowiedzi) pójdzie gdzie indziej. Atak zwany *smurf* używa sfabrykowanego adresu źródła. Z reguły odbywa się to w taki sposób, że następuje wysłanie pakietu do hosta, który nie przyjmuje pakietu z adresem źródłowym hosta. Najbardziej popularna metoda takiego ataku to użycie adresu rozgłaszania jako adresu źródłowego

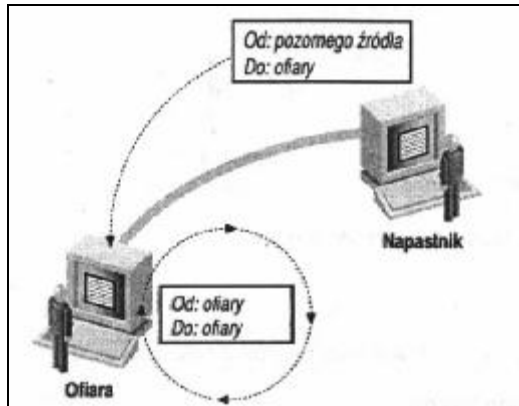
Rys. 15 Napastnik używa sfabrykowanych pakietów do zaatakowania trzeciego komputera.



Źródło: *Internet Firewalls* wyd. RM str.91

Inną odmianą tego ataku jest atak *lant* polegający na wysłaniu pakietu, którego źródło jest takie samo jak cel, co powoduje blokowanie maszyny (zapętlenie).

Rys. 16 Napastnik używa zapętlonych sfabrykowanych pakietów.



Źródło: *Internet Firewalls* wyd. RM str.92

4.4 Przechwytywanie pakietów

W celu zebrania informacji używa się podsłuchiwanie (*packet sniffing*) przepływających pakietów. Najczęściej używanie niezaszyfrowanego połączenia w sieciach typu Ethernet o topologii szyny lub sieć o okablowaniu 10 - base T umożliwia podsłuchiwanie przez każdy host, który podłączony jest do danego segmentu sieci. Spowodowane jest to tym, że pakiety są „kierowane do wszystkich”. Istnieje możliwość ograniczenia tego procederu poprzez instalację urządzeń sieciowych; rutery, switchy itp. Przechwytywanie pakietów używane jest także w celu analizie ataków przez programy *IDS* (Snort, tcpdump, Dragon).

V. Technologie stosowane w firewallach

Ściany ogniowe działają w oparciu o trzy podstawowe mechanizmy:

1. Filtrowanie pakietów (*packet filtering*): sposób działania polega na tym, że odrzucane są pakiety TCP/IP z nieautoryzowanych hostów i próby połączenia z nieautoryzowanymi usługami.

2. Translacja adresów sieciowych (NAT) (*Network Address Translation*): polega na dokonywaniu zmiany adresu *IP* hosta wewnętrznego w celu ukrycia go przed zewnętrznym monitorowaniem. Mechanizm ten jest również nazywany maskowaniem adresu *IP* (*IP masquerading*).

3. Usługi proxy: ściana ogniowa może dokonywać połączenia na poziomie aplikacji w imieniu wewnętrznego hosta, przerywane jest wtedy połączenie na poziomie sieciowej pomiędzy hostami wewnętrznymi i zewnętrznymi.

Dodatkowo ściany ogniowe mogą obsługiwać:

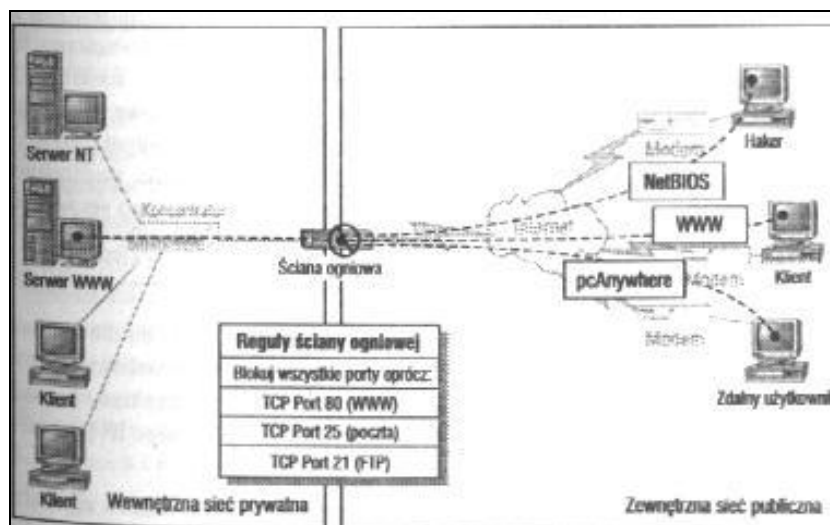
4. Szyfrowane uwierzytelnianie (*encrypted authentication*) użytkownicy sieci publicznej muszą udowodnić swoją tożsamość wobec ściany ogniowej zanim uzyskają dostęp do sieci wewnętrznej.

5. Szyfrowane tunelowanie (*encrypted tunneling*) pozwala ono ustanowić bezpieczne połączenie między dwiema prywatnymi sieciami za pośrednictwem publicznego medium typu Internet. Dzięki temu dwie fizycznie rozdzielone sieci mogą użyć do komunikowania się Internetu zamiast linii dzierżawionej. Tunelowanie jest również nazywane wirtualnymi sieciami prywatnymi VPN (*Virtual Private Networking*).

5.1 Flirty pakietów

Pierwsze ściany ogniowe były to flirty pakietów. Zasada działania polega na porównaniu pakietów protokołów sieciowych (*IP*) i transportowych (*TCP*) z regułami zawartymi w bazie danych, po to, aby dalej przekazać te pakiety, które odpowiadają kryteriom określonym w regułach.

Rys. 17 Przykładowe flirtowanie połączenia Internetowego.



Źródło: Ściany ogniowe wyd. Mikom str.156

Podstawowe filtrowanie może opierać się na:

1. Numer protokołu *IP*.
2. Numer portu *TCP*.
3. Numer portu *UDP*.
4. Na podstawie usługi.

Filtrowanie takie nic nie robi z danymi: nie podejmuje decyzji opartych o ich zawartość. Proste filtrowanie opiera się na numerach portów i stwierdzeniu czy dane są prawidłowe dla używanego protokołu.

5.2 Dynamiczne (stanowe) filtrowanie pakietów

Polega na przechowywaniu stanu i/lub sprawdzeniu protokołu. Najprościej można go opisać na zasadzie reguł:

„Pozwól wejść pakietom UDP tylko wtedy, gdy stanowią odpowiedź na widziane niedawno pakiety wychodzące UDP.

lub

Akceptuj pakiety TCP z ustawionym SYN tylko wtedy, gdy stanowią część nawiązanego połączenia TCP.”¹⁹

Reguły te nazywamy stanowym filtrowaniem pakietów (*stateful packet filtering*) ponieważ przechowywany jest stan transakcji. Jednakże filtrowanie takie ma pewne wady. Ruter musi pamiętać informacje o stanie, co powoduje zwiększenie obciążenia, ponad to może powodować ataki DoS. Dodatkowo nie zawsze mamy gwarancję, że pakiet z odpowiedzią otrzymamy (protokół UDP).

5.3 Filtrowanie na podstawie adresu

Jest to najprostsze filtrowanie, pozwala na ograniczenie przepływu pakietów na podstawie adresów źródłowych i/lub docelowych pakietów nie sprawdzając jakich protokołów użyto. Jednak filtrowanie takie ma wady, gdyż adres źródłowy można zawsze sfałszować (fałszowanie adresu źródłowego i atak „człowiek w środku”). Najprościej podać się za hosta uprawnionego do połączenia jednak, aby to zrobić musimy go najpierw unieruchomić. Najprościej go po prostu zawiesić, zalać pakietami, zakłócić trasowanie itp.

Możemy zadać sobie pytanie:

Co się dzieje kiedy pakiet zostanie odrzucony ?.

¹⁹ Internet Firewalls wyd. RM str. 158

„Zostaje wysłana odpowiedź ICMP, najbardziej istotne odpowiedzi to:

1. Kod „*destination unreachable*” (cel nieosiągalny) – w szczególności „*host unreachable*” i „*network unreachable*” (host/sieć nieosiągalna).
2. Kod „*destination administratively unreachable*” (cel nieosiągalny z powodów administracyjnych)- w szczególności kody „*host administratively unreachable*” i „*network administratively unreachable*” (host/sieć nieosiągalna z powodów administracyjnych).

Pierwsza para kodów została zaprojektowana w celu analizy problemów sieciowych. Drugi zestaw został dodany aby umożliwić filtrom pakietów przekazanie informacji kiedy pakiet został odrzucony. Jednak informacje takie pozwalają na ujawnienie dla napastnika, iż na drodze znajduje się np. ściana ogniowa.”²⁰

Jeżeli już ustawiamy pewne reguły filtrowania używajmy zawsze adresów *IP* hostów gdyż może się okazać, że np. serwer DNS zmieni nazwy hostów i nasze reguły przestaną działać lub nie prawidłowo filtrować będą pakiety .

5.4 Maskowanie adresu IP

Translacja adresów sieciowych NAT inaczej też maskowanie adresów *IP* rozwiązuje problem ukrywania wewnętrznych hostów. NAT opiera się na fakcie, iż tylko jeden host widoczny jest w sieci zewnętrznej, jednocześnie pełni on rolę pośrednika dla hostów wewnętrznych i w ich „imieniu” żąda podania informacji. Ponadto ukrywa on adresy wewnętrzne poprzez przyznanie mu własnego adresu zewnętrznego, identyfikacja opiera się na zapamiętaniu numeru portu, na którym przesyłana jest informacja dla danego hosta. Dodatkowo używanie NAT pozwala nam na przyłączenie większej liczby maszyn posiadając jeden adres *IP* wyjściowy. Z punktu widzenia Internetu cały ruch sieci użytkownika wydaje się pochodzić z jednego komputera. Dodatkowo używając NAT możemy korzystać w sieci wewnętrznej z dowolnej puli adresów. Wadą korzystania z mechanizmu *NAT* jest to, że implementowany jest on na poziomie warstwy transportowej. Oznacza to, że informacja ukryta jest w części danych pakietu *TCP/IP* może być przesłana do usług wyższej

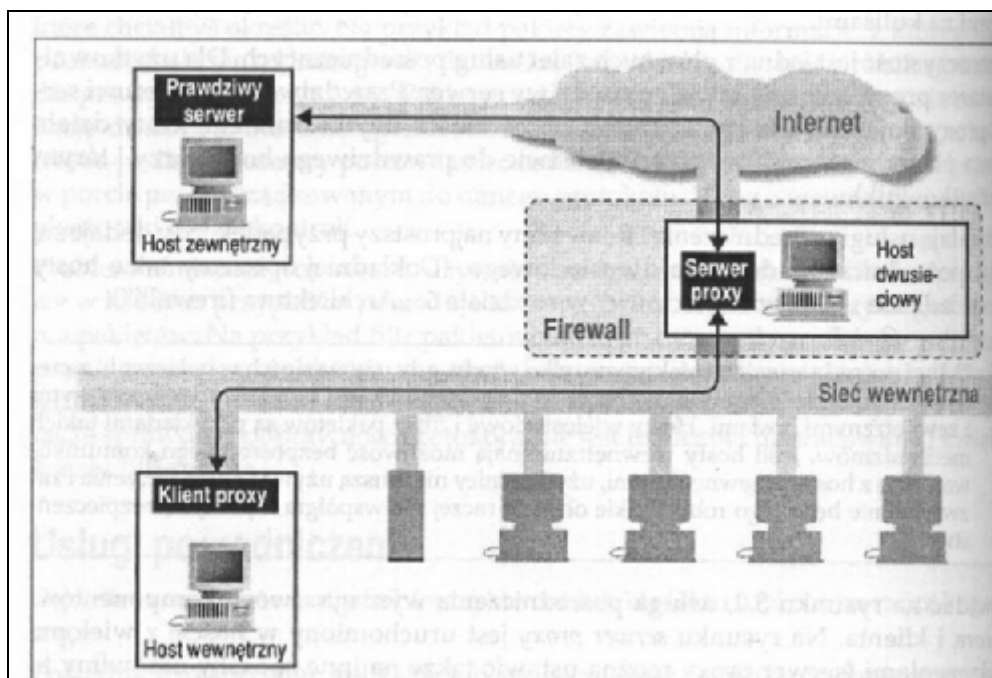
²⁰ Internet Firewalls wyd. RM str.164

warstwy i wykorzystana do eksploatacji słabości w innej warstwie lub komunikowania się z koniem trojańskim.

5.5 Proxy

NAT rozwiązuje wiele problemów związanych z bezpośrednim połączeniem z Internetem, ale nadal nie ogranicza całkowicie przepływu datagramów przez ścianę ogniową. Haker może przechwycić połączenie TCP (*connection hacking*) lub je sfalszować (*spoofing*). Aby uniemożliwić takie działania hakera stosujemy proxy aplikacyjne (*application proxy*). Proxy przyjmuje próbę połączenia z zewnętrznymi serwerami i następnie w imieniu klienta wykonuje połączenie do serwera docelowego. Aby dokładnie zabezpieczyć sieć, proxy musi obsługiwać filtr pakietów, który między innymi zabezpiecza przed atakami warstwy sieciowej np. typu DoS . Jako proxy uniwersalne można tu podać SOCKS, TIS, SQUID.

Rys. 18 Usługi pośredniczenia realizowane na hoście dwusiecznym (Proxy).



Źródło: *Internet Firewalls* wyd. RM str.104

5.6 Szyfrowane tunele

Zwane też, wirtualnymi sieciami prywatnymi (*Virtual Private Network*) pozwalają na bezpieczne połączenie za pomocą Internetu dwóch fizycznie rozdzielonych sieci bez narażania danych na monitorowanie. Same szyfrowane tunele mogą być przedmiotem ataków , jak próby zmiany trasy (*redirection*), inicjalizacja fałszywego połączenia (*spoofed connection initiation*) i inne nadużycia możliwe do popełnienia, gdy tunel jest już otwarty.

5.7 Szyfrowane uwierzytelnianie

Szyfrowane uwierzytelnianie (*encrypted authentication*) pozwala zewnętrznym użytkownikom z Internetu udowodnić ścianie ogniowej, że mają autoryzację czyli mogą otworzyć połączenie przez ścianę ogniową.

VI. Systemy Uniksowe

Przed konfiguracją firewalli w systemie uniksowych musimy uwzględnić następujące punkty:

1. Określić swoje potrzeby
2. Ocenić dostępne produkty
3. Połączyć wszystko w działający firewall

Ad 1.

Przed przystąpieniem do zakupu czy konstrukcji ściany ogniowej musimy uwzględnić:

1. Jakie zadania będzie wykonywał nasz firewall – najkrócej mówiąc jak bardzo będziemy chcieli zabezpieczyć naszą sieć?
2. Jakie usługi będziemy oferować i komu?
3. Jak bardzo wykorzystywana jest sieć?
4. Jak duża niezawodność jest nam potrzebna?
5. Jaki posiadamy ograniczenia?
6. Jaki posiadamy budżet?
7. Jakie środowisko jest najbardziej znane?

Ad 2

1. Skalowalność- czy pozwala na rozbudowę.
2. Raportowanie
3. Cena.
4. Zarządzanie i konfiguracja.

Ad 3

1. Gdzie i jak będą przesyłane raporty?
2. Odbudowa systemu w razie awarii czy ataku.
3. Czy potrzebne są dodatkowe usługi (uwierzytelnianie?).
4. Sposób dostępu (zdalny, fizyczny).
5. Gdzie będą kierowane sygnały alarmowe?

W systemach uniksowych mamy do dyspozycji dwa pakiety są to w FreeBSD IP-Filter i w OpenBSD PF składnia ich jest podobna dlatego omówię IP-Filtra na końcu wskazując różnicę między jednym a drugim.

6.1 IP-Filter²¹

W IP-Filter reguły są przetwarzane z góry na dół a sposób interpretacji reguł jest inny. IP-Filter utrzymuje flagę dla aż do skończenia przetwarzania wszystkich reguł i decyzje podejmuje na podstawie ostatniej pasującej reguły. Wygląda to w następujący sposób:

```
block in all
pass in all
```

Na początku jest sprawdzana pierwsza reguła, która mówi, że wszystkie pakiety są blokowane ustawiana jest flaga oznaczając, że na razie pakiety są blokowane. Następnie jest przetwarzana druga reguła, która nakazuje przepuścić pakiet. Tak więc ustawiana jest flaga, że pakiet jest przepuszczany, po której sprawdzana jest następna reguła jeżeli jej nie ma pakiet jest przepuszczany. Możemy więc zauważyć, że wygrywa ostatnia reguła. Istnieje także możliwość takiego zdefiniowania reguły, żeby sprawdzana była tylko jedna a pozostałe były pominięte, służy do tego polecenie `quick`, które powoduje, że tylko jedna reguła będzie sprawdzana a pozostałe zostaną pominięte.

```
block in quick
pass in all
```

W takim wypadku wszystkie pakiety będą blokowane a pozostałe reguły nie zostaną uwzględnione w przetwarzaniu pakietu przez IP-Filtra, następnych reguł mogłoby po prostu nie być.

²¹ Napisane na podstawie „Ściany ogniowe oparte o IP- Filter”

<http://mr0vka.eu.org/tlumaczenia/ipf.html>

IP-Filter jak każdy firewall posiada możliwość blokowania pakietów przychodzących z danego adresu *IP*.

```
block in quick from 192.168.0.0/16 to any
pass in all
```

W tym wypadku wszystkie pakiety przychodzące z adresów 192.168.0.0 /16 będą blokowane zaś pozostałe będą przepuszczane.

Możemy też filtrować pakiety nie tylko z jakiego adresu przychodzą ale również na podstawie interfejsu. Jeżeli będziemy chcieli otrzymywać pakiety na interfejs tun0 ale nie z adresów 192.168.0.0 /16.

```
block in quick on tun0 from 192.168.0.0 /16 to any
pass in all
```

Budowę firewalla zaczynamy od blokady pakietów przychodzących z adresów prywatnych.

```
block in quick on tun0 from 192.168.0.0/16 to any
block in quick on tun0 from 172.16.0.0/12 to any
block in quick on tun0 from 10.0.0.0/8 to any
block in quick on tun0 from 127.0.0.0/8 to any
block in quick on tun0 from 0.0.0.0/8 to any
block in quick on tun0 from 169.254.0.0/16 to any
block in quick on tun0 from 192.0.2.0/24 to any
block in quick on tun0 from 204.152.64.0/23 to any
block in quick on tun0 from 224.0.0.0/3 to any
pass in all
```

Brakuje nam jeszcze ruchu wyjściowego ponieważ dotychczas określiliśmy filtry tylko dla ruchu przychodzącego czyli *in* (*input*). Dlatego należy rozwinąć nasze reguły o ruch wychodzący. Możemy zatem blokować pakiety z fałszowanym adresem źródłowym, które nie powinien pojawić się poza naszą siecią wewnętrzną. Możemy

zatem ograniczyć ruch wychodzący tylko do pakietów których *IP* ma adres z zakresu sieci klasy A 10.0.0.1 /8 pozostały ruch zatem blokujemy.

```
pass out quick on tun0 from 20.20.20.0/24 to any
block out quick on tun0 from any to any
```

IP- Filter umożliwia nam zapisywanie informacji jaki ruch się odbywa, jakie pakiety przechodzą przez niego z jakich adresów. Informacje takie mogą być potrzebne administratorowi w celu analizy problemów np. skanowanie sieci.

```
block in log quick on tun0 from any to 10.0.20.0/8
block in log quick on tun0 from any to
10.255.255.255/8
```

Wpis ten ogranicza nam między innymi ataki smurf

W IP-Filter w celu kontroli poszczególnych protokołów używamy następującej składni

```
block in log quick on tun0 proto icmp from any to any
```

Jednak nie możemy blokować wszystkich komunikatów ICMP ponieważ w przypadku problemów z naszą siecią nie dostaniemy żadnej informacji o fakcie np. że dana sieć jest niedostępna. Możemy zatem wybrać, które typy komunikatów ICMP przepuszczamy, a które zostaną odrzucone służy do tego następująca reguła.

```
pass in quick on tun0 proto icmp from any to
10.0.0.0/0 icmp-type 0
pass in quick on tun0 proto icmp from any to 10.00.0.0/8
icmp-type 11
```

W tym przypadku przepuszczone zostaną pakiety ICMP numer 0 i 11 pozostałe zaś możemy odrzucić. Jednak przy użyciu `quick` musimy zdefiniować firewallowi, że pozostałe pakiety ma odrzucić.

```
block in log quick on tun0 proto icmp from any to any
```

Filtrowanie pakietów kierowanych do lub z danego portu

```
block in log quick on tun0 proto tcp from any to
10.0.0.0/24 port = 139
block in log quick on tun0 proto tcp/udp from any to
10.0.0.0/24 port = 53
block in log quick on tun0 proto tcp from any to
10.0.0.0/24 port = 23
```

W tym przypadku blokowane są pakiety przychodzące na interfejs tun0 z sieci 10.0.0.0 /24 na port 139. 23. 53. Jednak co się stanie jeżeli porty przyznawane są w sposób dynamiczny a nie statyczny, nie możemy zatem odwołać się do numeru portu, ponieważ w tej chwili go nie znamy. Musimy zatem zastosować politykę domyślnego zakazu wszystkiego co do nas dociera a dopiero później przepuszczać pakiety, które nas interesują.

```
block in all
block out all
pass in quick on tun0 proto udp from any to 10.0.0.0/16
port = 53
```

Na początku blokujemy wszystko aby później zezwolić na ruch pakietów do hosta 10.0.0.0/16 na port 53 (*DNS*) pozostałe pakiety będą odrzucone.

W IP-Filter tak jak w innych firewallach z rodziny uniksowych istnieje możliwość badania stanu danego połączenia na podstawie analizy flag ustawionych w nagłówku IP (SYN, ACK, FIN itp.) umożliwia to badanie połączeń protokołów TCP, UDP, ICMP służy do tego funkcja `keep state`. Pozwala nam to filtrować pakiety w następujący sposób: wiemy, że każda sesja ma swój początek środek i koniec, czyli tak naprawdę tylko początek połączenia będzie przechodził przez wszystkie reguły

firewalla zaś pozostałe będą przepuszczane, ponieważ należą do już nawiązanego połączenia. Tak więc jeżeli pakiet, który nawiązał połączenie ma prawo przejść to dalsze pakiety też mają takie prawo. W tym przypadku mamy do czynienia z firewallem stanowym, który bada stan połączenia. W celu rozwinięcia naszego firewalla możemy badać ustawione flagi w nagłówkach *IP*, możemy przepuszczać ruch tylko np. z ustawioną flagą SYN, określamy to za pomocą `flags np.`

```
pass out quick on tun0 proto tcp from any to any
flags S keep state
```

```
block in quick all
block out quick all
```

Akceptujemy ruch do hosta o numerze *IP* 10.0.0.0/8 z ustawioną flagą SYN zaś pozostałe pakiety blokujemy. Dodatkowo chronimy naszą sieć przed skanowaniem z ustawioną np. flagą FIN gdyż tylko te pakiety będą miały możliwość połączenia, jeżeli będą miały ustawioną odpowiednią flagę. Jednak jeżeli otrzymamy pakiet sfragmentowany musimy wskazać, że pakiety takie mogą przejść przez naszą ścianę ogniową za pomocą `keep frags`

```
pass in quick on tun0 proto tcp from any to
10.0.0.0/8
port = 23 flags S keep frags keep state
```

```
pass out quick on tun0 proto tcp from any to any
flags S keep state keep frags
```

```
block in log quick all
block out log quick all
```

Dodatkowo wszystkie pozostałe pakiety logujemy.

Jednak nie jest to odpowiednie wyjście, gdyż w przypadku, kiedy nie poinformujemy nadawcy, że pakiet został odrzucony może on bezskutecznie próbować połączyć się z

nami. W takiej sytuacji należy wysłać mu informację o tym poprzez wysłanie pakietu z flagą RST.

```
block return-rst in log proto tcp from any to
10.0.0.0/8 port = 23
block in log quick on tun0
pass in all
```

Jednak musimy też pamiętać, że istnieją skanery (*nmap*, *firewalk*), które umożliwiają skanowanie hostów poza firewallami za pomocą skanowania *stealth*. Jeśli nie chcemy ujawniać, że mamy jakąkolwiek ścianę ogniową musimy tak skonfigurować naszą ścianę ogniową aby nie zmieniała *TTL*, którą to wartość wykorzystują między innymi program tracert. W takim wypadku stosując *fastroute* spowodujemy, że pakiet nie zostanie przekazany do stosu *IP* w celu wykonania routingu co automatycznie zmieniłoby wartość *TTL*.

```
block in quick on x10 fastroute proto udp from any to any
port 33434 >< 33465
```

W przykładzie powyżej pakiet zostanie od razu umieszczony na interfejsie *x10* przez IP-Filter nie zmieniając żadnych jego parametrów. Oczywiście zostanie sprawdzona tablica routingu, ale routing nie wykona jądro lecz sam IP-Filter. Nie jest to wydajna technika, ale jeżeli zależy nam na nie wykrywalności to należy ją zastosować.

6.1.1 NAT i proxy

NAT w IP - Filter wykonuje się za pomocą polecenia ;

```
map tun0 192.168.0.0/16 -> 213.200.20.1/32
```

Musimy jednak inaczej napisać reguły w przypadku kiedy mamy przyznany adres za pomocą serwera DHCP. W IP - Filter nie jest to problemem gdyż możemy użyć następującej składni.

```
map ppp0 192.168.0.0/16 -> 0/32
```

W takim przypadku NAT sam sprawdzi, jaki adres został mu przydzielony, jeżeli dzierżawa tego adresu już wygasła. Jednak adres *IP* to nie wszystko, co się stanie jeżeli dwie usługi odwołają się do tego samego portu, nastąpi wtedy konflikt. Możemy zatem wskazać jaki zakres portów ma używać NAT poprzez dodanie słowa `portmap`.

```
map tun0 192.168.0.0/16 -> 0/32 portmap tcp/udp
20000:30000
```

Wszystkie połączenia, które używają protokołu *TCP* lub *UDP* używają zakresu portów od 20000:30000. Jednak najczęstsze zastosowanie NAT umożliwia połączenia z sieciami WAN (Internet) za pomocą jednego adresu *IP*.

```
map ppp0 192.168.0.0/16 -> 20.20.20.0/24 portmap
tcp/udp 20000:60000
```

Jednak czasami chcemy aby z zewnątrz nasze połączenia wyglądały jakby pochodziły z jednego adresu *IP* dlatego zastosujemy następujące polecenie.

```
map-block ppp0 192.168.0.0/16 -> 212.160.0.0/32
```

Możemy również zmienić tak ustawienia aby tylko jedna maszyna z sieci wewnętrznej miała zmieniany adres :

```
bimap ppp0 192.168.1.1/32 -> 212.160.0.0/32
```

W przeciwieństwie do `map` , `bimap` analizuje również stan połączenia oraz mapuje połączenie dwukierunkowo. Zastanówmy się nad następującym problemem, z którym często możemy się spotkać jeżeli posiadamy serwer WWW. Z reguły pracuje on na porcie 80 czyli uprzywilejowanym a nie chcemy aby tak było. Należy zatem

przekierować wszystkie połączenia z portu 80 na 8080 za pomocą polecenia `rdr` i następującej regułki.

```
rdr ppp0 213.160.0.0/32 port 80 -> 192.168.0.5 port
8080
```

Jednak nie możemy zrobić następującej rzeczy;

```
rdr ppp0 192.168.0.1/24 port 80 -> 192.168.0.5 port 80
```

ponieważ komputery 1 i 5 znajdują się w tym samym fragmencie sieci.

Ale częstym zastosowaniem jest uruchomienie serwera proxy w sieci wewnętrznej po to aby użytkownicy nie mieli możliwości połączenia się siecią zewnętrzną (Internet)ale łączyli się z wykorzystaniem serwerów proxy.

```
rdr x10 0.0.0.0/0 port 80 -> 168.0.0.1 port 8080
```

Dzięki temu wszystkie pakiety pochodzące z interfejsu `x10` (sieć wewnętrzna LAN) na port 80 ma być przekierowany do hosta numerze IP 168.0.0.1 na port 8080.

W IP-Filtrze jest trochę inaczej zorganizowana obsługa NAT, chodzi tu o pliki konfiguracyjne, z reguły chcemy mieć możliwość dokonywania NAT i filtrowania ruchu na ścianie ogniowej. Za NAT odpowiada plik `ipnat.rules`, zaś za reguły firewalla odpowiada `ipf.rules`. Tak więc jeżeli chcemy udostępnić serwer WWW w Internecie musimy dokonać następującego wpisu w `ipnat.rules`

```
rdr ppp0 200.20.20.5/32 port 80 -> 192.168.0.5
port 8080
```

zaś w `ipf.rules`

```
pass in on ppp0 proto tcp from 200.20.20.5/32 to
192.168.0.5/32 port = 8080 flags S keep state
```

wydawać by się mogło to trochę dziwne ale najpierw jest odczytywany plik `ipnat.rules` dokonywana jest translacja zaś później następuje sprawdzanie reguł

firewalla. IP-Filter pozwala na realizację serwerów proxy musimy jednak wcześniej powiadomić o tym IP-Filter, że będzie miał do czynienia z serwerem proxy. Służy do tego polecenie proxy.

```
map ppp0 192.168.1.0/24 -> 20.20.20.1/32 proxy port
ftp ftp/tcp
```

W tym wypadku najpierw używamy proxy zaś później nastąpi przemapowanie połączenia, w tym wypadku będzie to sesja FTP.

6.1.2 Zmiany reguł IP-Filtra

Reguły dotyczące IP- Filtra mogą znajdować się gdziekolwiek ale najczęściej znajdują się w następujących plikach

```
/etc/ipf.rules
/usr/local/etc/ipf.rules
/etc/opt/ipf/ipf.rules
```

IP-Filter pracuje z dwoma regułami: aktywną i nieaktywną, pozwala nam to na pracę z danymi regułami nie zmieniając przy tym reguł działającej ściany ogniowej. Ładowanie reguł jako aktywne odbywa się za pomocą parametru `ipf -s`. Usuwanie reguły odbywa się za pomocą `ipf -r` ale nie jest zalecane. Na co dzień zalecane jest stosowanie polecenia `ipf -F` czyli usunięcie wszystkich reguł i załadowanie nowych za pomocą polecenia `ipf -Fa -f /etc/ipf.rules`.

6.1.3 Zmiany reguł IP-Filtra dla NAT

Tak jak do ładowania reguł dla IP-Filtra mamy narzędzie `ipf` tak w celu załadowania reguł dla NAT musimy posłużyć się `ipnat`. Pliki konfiguracyjne przeważnie przechowywane są w plikach

```
/etc/ipnat.rules,
/usr/local/etc/ipnat.rules
/etc/opt/ipf/ipnat.rules.
```

Składnia polecenia `ipnat` jest podobna do `ipf` jedyna różnica polega na tym, że zamiast używać polecenia `ipnat -r` do usuwania reguł używa się `ipnat -C` lecz nie usuwa to mapowanych połączeń dlatego musimy używać `ipnat -F` aby usunąć mapowane połączenia. Możemy sprawdzić jakie są aktualne reguły NAT i mapowania jeżeli wpisujemy `ipnat -l`. Ładowanie reguł najlepiej wykonać wpisując komendę `ipnat -CF -f /etc/ipnat.rules`.

6.1.4 Dodatkowe narzędzia dla IP-Filter

W celu sprawdzania stanu działania naszej ściany ogniowej możemy użyć kilku narzędzi, które pozwalają na dokładne przeanalizowanie co się dzieje z całym ruchem, który przechodzi przez firewalla są to :

1. `Ipfstat`.
2. `Ipmon`

Ad.1

`Ipfstat` wyświetla tabelę interesujących nas danych dotyczących tego, jak ściana ogniowa daje sobie radę z ruchem, który przez nią przechodzi. Między innymi są to: ilość pakietów, które zostały przepuszczone i zablokowane, czy zostały zalogowane czy nie, ile jest aktywnych pozycji w liście stanów i tak dalej. Przykład po uruchomieniu narzędzia:

```
# ipfstat
  input packets:          blocked 99286 passed 1255609
nomatch 14686 counted 0
  output packets:        blocked 4200 passed 1284345
nomatch 14687 counted 0
  input packets logged:   blocked 99286 passed 0
  output packets logged:  blocked 0 passed 0
  packets logged:         input 0 output 0
  log failures:           input 3898 output 0
  fragment state(in):     kept 0 lost 0
  fragment state(out):    kept 0 lost 0
```

```

packet state(in):      kept 169364      lost 0
packet state(out):    kept 431395      lost 0
ICMP replies:        0          TCP RSTs sent: 0
Result cache hits(in): 1215208 (out): 1098963
IN Pullups succeeded: 2          failed: 0
OUT Pullups succeeded: 0          failed: 0
Fastroute successes: 0          failures:      0
TCP cksum fails(in): 0          (out): 0
Packet log flags set: (0)      none

```

Ipstat może również pokazać aktualną listę reguł. Użycie flagi `-i` lub `-o` pokaże listę reguł dla odpowiednio pakietów wchodzących i wychodzących. Dodanie opcji `-h` dodaje też trochę informacji, podając również 'licznik trafień' dla każdej reguły. Na przykład:

```

# ipfstat -ho
2451423 pass out on xl0 from any to any
354727 block out on ppp0 from any to any
430918 pass out quick on ppp0 proto tcp/udp from
20.20.20.0/24 to any keep state keep frags

```

Z przykładu powyżej widać, że prawdopodobnie dzieje się coś nienormalnego, ponieważ mamy bardzo dużo zablokowanych pakietów wychodzących, nawet przy użyciu reguły `pass out`. Coś może wymagać ewentualnej zmiany, albo po prostu firewall pracuje bez zarzutu. Ipstat nie może powiedzieć czy reguły są dobre czy złe, może tylko powiedzieć co się dzieje przy użyciu danych reguł.

By dalej zmieniać swoje reguły, możemy użyć opcji `-n` która pokaże numery reguł przy każdej z nich:

```

# ipfstat -on
@1 pass out on xl0 from any to any
@2 block out on ppp0 from any to any
@3 pass out quick on ppp0 proto tcp/udp from
20.20.20.0/24 to any keep state keep frags

```

Możemy również dokonać zrzutu tabeli stanów. Wykonuje się to dodając opcję `-s`:

```
# ipfstat -s
281458 TCP
319349 UDP
0 ICMP
19780145 hits
5723648 misses
0 maximum
0 no memory
1 active
319349 expired
281419 closed
100.100.100.1 -> 20.20.20.1 ttl 864000 pass 20490 pr 6 state
4/4
pkts 196 bytes 17394          987 -> 22 585538471:2213225493
16592:16500
pass in log quick keep state
pkt_flags & b = 2,          pkt_options & ffffffff = 0
pkt_security & ffff = 0, pkt_auth & ffff = 0
```

Widzimy, że mamy aktualnie jedno połączenie TCP. Widzimy dla tego połączenia, że jest ono w pełni nawiązane (`state 4/4` na końcu linijki). Możemy również odczytać, że wpis ten ma **czas życia** (ang. *time to live*) 240 godzin, co jest długim czasem, ale ustawianym domyślnie dla nawiązanej sesji TCP. Licznik ten jest zmniejszany z każdą sekundą gdy wpis nie jest używany, aż w końcu połączenie zostanie zerwane jeśli zostanie pozostawione bezczynnie. Licznik jest również zerowany na wartość 864000 za każdym razem gdy użyjemy wpisu więc połączenie nie zostanie zamknięte w trakcie jego używania. Możemy również odczytać, że przepuściliśmy przez to połączenie 196 pakietów składających się na około 17kB danych. Oprócz tego można odczytać porty po obu stronach - 987 i 22. Bardzo duże numery w drugiej linijce to numery sekwencyjne TCP wygenerowane dla tego połączenia, pomagające zabezpieczyć je przed wpuszczeniem dodatkowych, spreparowanych pakietów. Pokazane jest również okno TCP. Trzecia linia stanowi wynik reguły która została wygenerowana przez regułę `keep state` i pokazuje ona, że jest to połączenie przychodzące.

Ad.2

Ipfstat pozwala nam tylko na sprawdzenie aktualnego stanu systemu, a zwykle chcemy mieć również jakiś log, by oglądać wydarzenia dziejące się w czasie. Służy do tego ipmon. Jest on zdolny do sprawdzania logów pakietów (tworzonych przez słowo kluczowe log w regułach), logu tabeli stanów i logu NAT, lub dowolnej kombinacji tych trzech. Narzędzie to może pracować zarówno na pierwszym planie, lub jako demon, który loguje informacje do sysloga lub pliku. Jeśli chcielibyśmy zobaczyć listę stanów w akcji, używamy polecenia ipmon -o S

```
# ipmon -o S
01/08/1999 15:58:57.836053 STATE:NEW 100.100.100.1,53 ->
20.20.20.15,53 PR udp
01/08/1999 15:58:58.030815 STATE:NEW 20.20.20.15,123 ->
128.167.1.69,123 PR udp
01/08/1999 15:59:18.032174 STATE:NEW 20.20.20.15,123 ->
128.173.14.71,123 PR udp
01/08/1999 15:59:24.570107 STATE:EXPIRE 100.100.100.1,53 -
> 20.20.20.15,53 PR udp Pkts 4 Bytes 356
01/08/1999 16:03:51.754867 STATE:NEW 20.20.20.13,1019 ->
100.100.100.10,22 PR tcp
01/08/1999 16:04:03.070127 STATE:EXPIRE 20.20.20.13,1019 -
> 100.100.100.10,22 PR tcp Pkts 63 Bytes 4604
```

Możemy tu zauważyć np. zapytanie DNS z zewnętrznej maszyny do naszego serwera, Ipmon jest również w stanie pokazać nam jakie pakiety są logowane. Na przykład kiedy używamy stanów połączeń spotkamy się pakietami:

```
# ipmon -o I
15:57:33.803147 ppp0 @0:2 b 100.100.100.103,443 ->
20.20.20.10,4923 PR tcp len 20 1488 -A
```

Pierwsze pole to stempel czasu. Drugie to interfejs na którym wydarzyło się zdarzenie. Trzecie pole @0:2 to oznaczenie reguły która spowodowała zdarzenie (związane z regułą ipfstat -in) Jeśli chcemy wiedzieć co spowodowało zalogowanie pakietu powinniśmy obejrzeć regułę 2 w grupie 0. Czwarte pole małe

„b” mówi, że pakiet został zablokowany, i na razie możemy je ignorować, chyba że logujemy również pakiety, które przepuszczamy co spowoduje pokazanie się literki „p”. Piąte i szóste to adresy IP i mówią one skąd pakiet przyszedł i gdzie miał dotrzeć. Siódme [PR] i ósme pole to protokół, a dziewiąte długość pakietu. Ostatnia część “-A” to flagi, które były ustawione; ten pakiet ma ustawioną flagę ACK. Innym przykładem pakietu może być:

```
12:46:12.470951 x10 @0:1 S 20.20.20.254 ->
255.255.255.255 PR icmp len 20 9216 icmp 9/0
```

Jest to broadcast rozpoznawczy ICMP routera. Możemy to stwierdzić na podstawie typu ICMP: 9/0.

Na koniec, ipmon pozwala również na obejrzenie tabeli NAT:

```
# ipmon -o N
01/08/1999 05:30:02.466114 @2 NAT:RDR 20.20.20.253,113
<- -> 20.20.20.253,113 [100.100.100.13,45816]
01/08/1999 05:30:31.990037 @2 NAT:EXPIRE
20.20.20.253,113 <- -> 20.20.20.253,113
[100.100.100.13,45816] Pkts 10 Bytes 455
```

Możemy zauważyć przekierowanie do serwera identd port 113.

Na koniec musimy pamiętać, że aby uruchomić firewalla musimy ustawić odpowiednie parametry pracy jądra czyli:

IP Forwarding:

OpenBSD:

```
net.inet.ip.forwarding=1
```

FreeBSD:

```
net.inet.ip.forwarding=1
```

NetBSD:

```
net.inet.ip.forwarding=1
```

Solaris:

```
ndd -set /dev/ip ip_forwarding 1
```

Zmiany dotyczące ustawień portów:

OpenBSD:

```
net.inet.ip.portfirst = 25000
```

FreeBSD:

```
net.inet.ip.portrange.first = 25000
```

```
net.inet.ip.portrange.last = 49151
```

NetBSD:

```
net.inet.ip.anonportmin = 25000
```

```
net.inet.ip.anonportmax = 49151
```

Solaris:

```
ndd -set /dev/tcp tcp_smallest_anon_port 25000
```

```
ndd -set /dev/tcp tcp_largest_anon_port 65535
```

Inne wartości to :

OpenBSD:

```
net.inet.ip.sourceroute = 0
```

```
net.inet.ip.directed-broadcast = 0
```

FreeBSD:

```
net.inet.ip.sourceroute=0
```

```
net.ip.acceptsourceroute=0
```

NetBSD:

```
net.inet.ip.allowsrcrt=0  
net.inet.ip.forwsrcrt=0  
net.inet.ip.directed-broadcast=0  
net.inet.ip.redirect=0
```

Solaris:

```
ndd -set /dev/ip ip_forward_directed_broadcasts 0  
ndd -set /dev/ip ip_forward_src_routed 0  
ndd -set /dev/ip ip_respond_to_echo_broadcast 0
```

Dodatkowo, FreeBSD ma pewne dodatkowe zmienne sysctl:

```
net.inet.ipf.fr_flags: 0  
net.inet.ipf.fr_pass: 514  
net.inet.ipf.fr_active: 0  
net.inet.ipf.fr_tcpidletimeout: 864000  
net.inet.ipf.fr_tcpclosewait: 60  
net.inet.ipf.fr_tcpplastack: 20  
net.inet.ipf.fr_tcptimeout: 120  
net.inet.ipf.fr_tcpclosed: 1  
net.inet.ipf.fr_udptimeout: 120  
net.inet.ipf.fr_icmptimeout: 120  
net.inet.ipf.fr_defnatage: 1200  
net.inet.ipf.fr_ipfrttl: 120  
net.inet.ipf.ipl_unreach: 13  
net.inet.ipf.ipl_initiated: 1  
net.inet.ipf.fr_authsize: 32  
net.inet.ipf.fr_authused: 0  
net.inet.ipf.fr_defaultauthage: 600
```

Różnice między IP-Filter a PF są niewielkie praktycznie żadne. Problem polega jedynie na interpretowaniu pewnych słów kluczowych np w PF nie ma słów kluczowych `head` i `group` a także PF używa mechanizmu pomijania w celu optymalizacji reguł.²²

6.2. Firewall oparty na systemie operacyjnym LINUXS

6.2.1 Przygotowanie Linuksa

Z pośród wielu pakietów Linuks oferuje wiele narzędzi, które umożliwiają użycie go jako firewalla. Istnieje możliwość łączenia ich ze sobą dodatkowo już na poziomie jądra. W Linuksie istnieje możliwość dodania filtrów pakietów:

1. `ipfw` .
2. `ipchains` (2.2.).
3. `netfilter` (2.2.4).

Dodatkowo w celu dodatkowej ochrony naszej sieci możemy użyć proxy :

1. SOCKS.
2. TIS.

Zaletą proxy jest fakt, że nie dokonują bezpośredniego routingu.

Narzędzia do zabezpieczenia firewalla

Dodatkowe narzędzia, które umożliwiają nam kontrolę bezpieczeństwa naszego firewalla nazywamy IDS (*Intrusion Detection System*). Możemy podzielić je na grupy :

- a) systemy wykrywające próby włamania w ich początkowej fazie
- b) systemy wykrywające naruszenie integralności serwera po włamaniu
- c) jako osobną grupę możemy zaliczyć analizatory logów

²² Zainteresowanych odsyłam do dokumentacji PF na stronach www.openbsd.org

Z pierwszej grupy najbardziej znane są produkty firmy *Psionic Software* znane jako projekt *Abacus* jest to zbiór narzędzi IDS. Do pakietu tego możemy zaliczyć:

1. Detektor skanów *PortSentry*, którego zadaniem jest wykrywanie prób ataków sieciowych i blokowanie ich poprzez wyłączenie atakowanej usługi, dynamiczną zmianę zasad filtracyjnych zapory sieciowej oraz dopisanie danego hosta do pliku i */etc/host.deny* lub wyłączenie atakowanej maszyny. Pakiet między innymi wykrywa skanowanie ukryte (*stealth scaninig*).
2. Detektor anomalii logowania *HostSentry* (*LAD Login Anomaly Detection*) informuje administratora o nietypowych zdarzeniach związanych z procesem logowania i pozwala na szybką reakcję w przypadku podejrzanego użytkownika. Pakiet *HostSentry* wyposażony jest w moduł uczący się, który analizuje typowe zachowania użytkowników i alarmuje w sytuacji, gdy występują anomalie. Program wykorzystuje logi *wtmp/utmp* w celu określenia typowej aktywności użytkownika. Monitorowanie aktywności użytkownika odbywa się podczas logowania jak i podczas wylogowywania, kiedy to sprawdzana jest np. czy historia poleceń użytkownika nie została skasowana lub przekierowana do */dev/null*.
3. *Logcheck/LogSentry* narzędzie do automatycznej analizy logów systemowych. Informacje o anomaliiach może wysłać do administratora za pomocą poczty elektronicznej.

Do drugiej grupy należy zaliczyć narzędzia do testowania integralności plików do najbardziej znanych należą:

1. *Tripwire*
2. *AIDE*
3. *Integrit*

Programy te służą do tworzenia sygnatur plików. Każdy plik opatrzony zostaje sygnaturą wygenerowaną za pomocą jednego lub kilku algorytmów: *SHA1*, *MD5*, *RMD160*, *tiger*, *HVAL*, *GOST*, *CRC32*.

Dodatkowo istnieje możliwość ochrony plików konfiguracyjnych, zaimplementowania w systemie plików Linuksa ext2 mówimy tu o ustawieniu flag dla danego pliku za pomocą zmiany praw do plików ustawianych za pomocą `chmod` (r- odczyt, w –zapis, x – uruchamianie).

Lepki bit musimy pamiętać, że jeżeli dany użytkownik posiada prawo do danego katalogu to może usunąć katalogi i pliki nie należące do niego nawet, jeżeli nie jest właścicielem tych plików, a nie posiada praw do tych plików. Aby temu zapobiec musimy ustawić, tzw. Lepki bit za pomocą polecenia `chmod + t`. Spowoduje to, że użytkownicy danego katalogu nie mogą usuwać katalogów i plików, których nie są właścicielami. Dodatkowo musimy pamiętać, że każdy użytkownik systemu, kiedy tworzy plik lub katalog nadawane mu są standardowe uprawnienia, są to:

- 664 dla plików
- 775 dla katalogów

Aby to zmienić należy zmienić wartość `umask`. Najbardziej restrykcyjne to `umask 777` jednak nie są to dobre ustawienia. Najlepszy wybór to ustawienie `umask 077`, najlepiej wartość tą wpisać do `./bash_profile`, aby za każdym razem była ustawiona po starcie systemu.

Atrybuty plików

Atrybuty plików są ustawiane dla większej ochrony i zabezpieczeń. Oprócz standardowych `rwX` w celu ustawienia tych atrybutów musimy użyć polecenia `chattr` flagi te może ustawić tylko `root` :

- A Nie aktualizuje pliku `atime`, który służy do ograniczenia wejścia/wyjścia dysku na komputerze typu laptop lub poprzez NFS. Atrybut ten nie jest obsługiwany przez starsze jądra Linuksa zwłaszcza starsze z serii 2.0
- a Otwiera plik w trybie tylko do dołączania
- c Plik z takim atrybutem jest automatycznie kompresowany na dysku przez jądro Linuksa
- d Oznacza plik jako nieprzeznaczony dla programu składającego
- i Plik z takim atrybutem nie może być modyfikowany, usuwany i przemianowany, nie można do niego utworzyć żadnego łącza ani zapisać żadnych danych
- s Gdy plik jest usuwany jego bloki są wyzerowywane i z powrotem zapisywane na dysk

- S Gdy plik jest modyfikowany, zmiany są synchronicznie zapisywane na dysk
- u Gdy plik jest usuwany jego zawartość jest zapisywana

Dodanie danej flagi odbywa się za pomocą znaku [+] a usunięci flagi [-].

Zanim przystąpimy do uruchomienia naszej ściany ogniowej należy skompilować jądro z odpowiednimi ustawieniami. Jednak większość obecnych dystrybucji Linuksa ma już wkompiłowaną obsługę ścian ogniowych czy też ruterów. Jeżeli tak nie jest należy włączyć przy kompilacji jądra następujące opcje:

1. Under General setup
 1. Turn Networking Support ON
2. Under Networking Options
 1. Turn Network firewalls ON
 2. Turn TCP/IP Networking ON
 3. Turn IP forwarding/gateway ON (UNLESS you wish to use IP filtering)
 4. Turn IP Firewalling ON
 5. Turn IP firewall packet logging ON (this is not required but it is a good idea)
 6. Turn IP: masquerading ON (I am not covering this subject here.)
 7. Turn IP: accounting ON
 8. Turn IP: tunneling OFF
 9. Turn IP: aliasing OFF
 10. Turn IP: PC/TCP compatibility mode OFF
 11. Turn IP: Reverse ARP OFF
 12. Turn Drop source routed frames ON
3. Under Network device support
 1. Turn Network device support ON
 2. Turn Dummy net driver support ON
 3. Turn Ethernet (10 or 100Mbit) ON
 4. Select your network card

Najlepiej przeczytać Kernel _HOWTO. Od jądra 2.4. dostępna jest specjalna opcja w sekcji *Network*, w której możemy skonfigurować dokładnie parametry jądra do pracy jako ściana ogniowa.

Dodatkowo musimy posiadać dwa interfejsy sieciowe, które muszą posiadać jeden adres wewnętrzny a drugi zewnętrzny. Przydzielamy im adresy sieciowe np. 192.168.0.1 adres interfejsu wewnętrznego i 213.212.140.1 adres interfejsu zewnętrznego (widzianego od strony Internetu)

Kiedy już mamy zainstalowanego Linuksa warto zastanowić się jakich usług będziemy potrzebować, a które są nam w ogóle nie potrzebne i możemy je wyłączyć. Za start usług, w Linuksie odpowiada demon *inetd* podczas startu systemu odczytywany jest plik konfiguracyjny (*/etc/inetd.conf* lub */etc/xinit.d*), który jest głównym plikiem konfiguracyjnym demona *inetd* tam też powinniśmy skierować swoje pierwsze kroki i dokładnie przeanalizować cały plik i wyłączyć niepotrzebne nam usługi stawiając znak # na początku linii z opisem danej usługi.

```
# local telnet services
[ # ] telnet-a stream tcp nowait root /usr/local/etc/tcpd
in.telnetd
```

lub też zmienić plik konfiguracyjny danej usługi poprzez dodanie wpisu. Spowoduje to, że usługa ta nie będzie ładowana przy starcie systemu np. */x.init.d/telnet*

```
disable="no"
```

Dodatkowo musimy pamiętać, że Linuks może być uruchomiony aż w siedmiu trybach pracy, w których uruchamiane są różne usługi (*/etc/inittab*) w zależności od trybu pracy systemu. Kontrolę tych usług możemy przeprowadzić za pomocą tzw. Wrappera. Działanie jego opiera się na fakcie, iż jeżeli użytkownik żąda danej usługi na określonym porcie np. telnet to demon *inetd* sprawdza w pliku */etc/services* jaka usługa jest uruchamiana na tym porcie i wtedy startuje odpowiedni program wyznaczony do jej obsługi w pliku */etc/inetd.conf*.

Dodatkowo istnieje możliwość kontroli startujących usług za pomocą wpisów w plikach */etc/host.deny* oraz */etc/host.allow*. Pierwszy z nich definiuje usługi, które są zabronione zaś drugi pozwala na korzystanie z danych usług. Należy pamiętać, że pierwszy jest czytany plik *host.allow* jeżeli nie zostanie znaleziony wpis to przeszukiwany jest *host.deny* w przypadku braku wpisu w tym pliku dana usługa jest dozwolona. Konfiguracja nie jest trudna, najlepiej zabronić wszystkich usług w pliku *host.deny* a później poszczególne usługi odblokować w *host.allow*. Wrapper do rejestrowania swoich komunikatów korzysta z właściwości *authpriv* demona *syslog*. Istnieje możliwość dodania dodatkowych parametrów, które mogą nam pomóc np. w identyfikacji połączeń:

„%a - adres IP klienta

%A - adres IP serwera

%c - wszystkie dostępne informacje o kliencie

%d - nazwa procesu demona usługi sieciowej

%h - nazwa hosta klienta (jeśli nie jest dostępna, używany jest adres IP)

%H - nazwa hosta serwera

%n - nazwa hosta klienta (jeżeli nie jest dostępna, używane jest słowo UNKNOWN. Jeżeli natomiast wyszukiwanie DNS nazwy hosta klienta i adresu IP nie zgadzają się, użyte zostaje słowo kluczowe PARANOID)

%N - nazwa hosta serwera

%p - ID procesu (PID) demona usługi sieciowej

%s - wszystkie dostępne informacje o serwerze

%u - nazwa użytkownika klienta (jeżeli nie jest dostępna, użyte zostanie słowo kluczowe UNKNOWN)²³

Za pomocą standardowej składni kontroli dostępu możemy zdefiniować prawie każdą potrzebną regułę. Jednak jeśli to nie wystarczy, można skorzystać z opcjonalnego rozszerzenia kontroli dostępu programu **Wrapper**. Jednak w takim przypadku należy skompilować własną wersję programu z włączoną opcją **PROCESS_OPTIONS**. Składnia języka kontroli dostępu zostanie wtedy rozszerzona i przedstawia się następująco:

”usługi : klienty : opcja : opcja ...

Do każdej reguły można zastosować wiele opcji, m.in.:

allow - przyznaje żadaną usługę (musi pojawiać się na końcu reguły!)

deny - odmawia przyznania żądanej usługi (również musi się pojawiać na końcu reguły)

spawn *polecenie powłoki* - wykonuje polecenie jako proces podrzędny

twist *polecenie powłoki* - wykonuje polecenie zamiast żądanej usługi

keepalive - wysyła do klienta komunikat sygnalizujący aktywność, a jeśli klient nie odpowiada, to połączenie zostaje zerwane

linger *sekundy* - określa, jak długo mają trwać próby dostarczenia danych po zamknięciu połączenia przez serwer

rfc931 [*czas oczekiwania*] - użycie protokołu IDENT w celu sprawdzenia nazwy użytkownika klienta. Czas definiuje, ile sekund serwer ma czekać na odpowiedź.

²³ www.linux.centromost.com.pl

banners *ścieżka* - wyświetla u klienta zawartość pliku z komunikatem. Ścieżka określa nazwę katalogu zawierającego pliki nagłówkowe (banner files). Wyświetlony zostaje ten plik, który posiada taką samą nazwę, jak proces demona danej usługi sieciowej.

nice [*numer*] - ustawia wartość nice dla procesu usługi. Jądro używa wartości nice do obliczenia priorytetów szeregowania.

umask *maska* - ustawia wartość umask dla plików tworzonych przez dany proces

user *użytkownik* [*grupa*] - uruchamia proces usługi sieciowej z określonym ID użytkownika i ID grupy, niezależnie od tego, co jest zdefiniowane w ident.conf

setenv *nazwa wartość* - ustawia zmienną środowiskową dla środowiska uruchomieniowego procesu.

Przykład:

```
in.rshd : ALL : spawn /usr/sbin/safe_finger -l @%c |  
/bin/mail -s %h%d root : DENY"24
```

Zdalny system próbuje uruchomić *rshd*, jednak w pierwszej kolejności uruchamiane jest polecenie *safe_finger*, a jego wyniki są wysyłane do konta administratora. Następnie połączenie jest zamykane. Wrapper daje duże możliwości zwiększenia bezpieczeństwa systemu. Należy jednak pamiętać, iż nawet korzystając z rozszerzonej składni **Wrappera**, możemy chronić tylko usługi uruchamiane przez *inetd* lub takie, które same z siebie czytają pliki *host.allow* i *host.deny*. Z Wrapperem bardzo dobrze współdziała PortSentry nasłuchuje on na portach *TCP/IP* badając próby nie autoryzowanego podłączenia się do systemu. Przy próbie skanowania zapisuje informację w logach poprzez demona *syslog* dodatkowo umożliwia automatyczny wpis

²⁴ www.linux.centromost.com.pl

do pliku */etc/host.deny*. Narzędzia te chronią tylko danego hosta a nie całą sieć, dlatego w celu dokładnej ochrony naszej sieci wewnętrznej musimy użyć *ipchains*. Jest to ściana ogniowa, która realizuje funkcje polegające na analizie połączeń opartych na *TCP/IP*.

Generalnie jądro systemu dzieli ruch firewalla na trzy kategorie i do każdej z nich stosuje inny filtr. Wchodzący ruch, zanim zostanie zaakceptowany, jest testowany według zasad wchodzącej ściany ogniowej. Wychodzący ruch przed wysłaniem jest testowany zgodnie z regułami wychodzącej ściany ogniowej natomiast ruch, który jest przekazywany poprzez system, jest testowany zgodnie z regułami dla przekazującej ściany ogniowej. Oprócz tych trzech standardowych kategorii, czyli: wchodząca ściana ogniowa (**input**), wychodząca ściana ogniowa (**output**) oraz przekazująca ściana ogniowa (**forward**) użytkownik może definiować także własne kategorie. Jądro dla każdej z kategorii utrzymuje listę reguł nazywanych łańcuchami.

Zalety takiego rozwiązania to:

1. Każdy pakiet wchodzący, przesyłany przez warstwy routujące Linuksa lub wychodzący, sprawdzany jest za pomocą reguły filtrowania pakietów.
2. Zintegrowanie *ipchains* z warstwą *IP* oraz kernelem powoduje szybkie działanie.
3. Możliwość stosowania NAT.
4. Możliwość implementacji VPN na zasadzie ściana ogniowa – ściana ogniowa lub ściana ogniowa – zdalny klient jest możliwa z pomocą dodatkowego modułu dostępnego w Linuksie.
5. Zarządzanie zdalne za pomocą zdalnego shella.

Wady:

1. Konfiguracja w trybie tekstowym.
2. Brak możliwości filtrowania zawartości określonych protokołów (WWW, SMTP). W celu rozwiązania tego problemu należy stosować dodatkowe filtry aplikacyjne (TIS FWTK, Jigsaw).

W celu utworzenia, usunięcia bądź dokonania innej operacji na łańcuchu, można

korzystać z następujących opcji polecenia *ipchains*:

”-**A** - dodaje regułę na koniec łańcucha

-**C** - sprawdza pakiet zgodnie z regułami w łańcuchu (używany do testowania definiowanych łańcuchów)

-**D** - usuwa wybraną regułę z łańcucha

-**F** - usuwa wszystkie reguły z łańcucha

-**I** - wstawia regułę do łańcucha

-**L** - wypisuje listę wszystkich reguł w łańcuchu

-**M** - definiuje parametry maskowania adresu lub wypisuje aktualne ustawienia

-**N** - tworzy zdefiniowany przez użytkownika łańcuch o określonej nazwie

-**P** - ustawia domyślną zasadę postępowania dla łańcucha

-**R** - zastępuje regułę w łańcuchu

-**S** - ustawia wartość czasu oczekiwania dla maskowania IP

-**X** - usuwa określony łańcuch

-**Z** - we wszystkich łańcuchach zeruje liczniki pakietów i bajtów”²⁵

Reguły ściany ogniowej składają się z filtra, do którego dopasowywane są pakiety. Gdy pakiet zostanie dopasowany, podejmowane jest działanie, które może być

²⁵ Linux IPCHAINS-HOWTO Bibliografia poz 12

albo standardową zasadą, albo może wskazywać na zdefiniowany przez użytkownika łańcuch reguł, w celu dalszego przetworzenia. „Standardowe zasady postępowania to:

accept - zezwala na przejście pakietu

reject - odrzuca pakiet zwracając do nadawcy komunikat o błędzie

deny - odrzuca pakiet, a do nadawcy nie jest wysyłany żaden komunikat

masq - maskuje pakiety w ten sposób, iż wyglądają, jakby pochodziły z lokalnego hosta

redirect - bez względu na przeznaczenie, pakiet jest dostarczany do portu w lokalnym hoście

return - powrót do łańcucha, który wywołał ten łańcuch. Powoduje to wyjście z łańcucha i użycie domyślnych zasad postępowania dla danego łańcucha.

Do konstruowania filtrów możemy użyć także parametrów polecenia **ipchains**:

-p protokół - definiuje protokół. Może mieć wartość numeryczną (taką, jak w pliku */etc/protocols*) lub może występować, jako słowo kluczowe, np. tcp, udp, icmp lub all.

-s adres [/maska] [port[:port]] - definiuje źródło pakietu. Adres może być nazwą hosta, nazwą sieciową lub numerem IP z opcjonalną maską adresową. Port może być nazwą lub numerem z pliku */etc/services*. Zakres portów może być określony jako port:port. Jeśli wartość portu nie jest określona, reguła dotyczy wszystkich portów.

-d adres [/maska] [port[:port]] - definiuje adres przeznaczenia pakietu.

-j cel - określa standardowe zasady postępowania lub zdefiniowany przez użytkownika łańcuch, do którego powinna być przekazana kontrola

input chain (łańcuch wejściowy):

To pierwszy łańcuch ściany ogniowej, na którym pakiet będzie testowany. Jeśli reguła nie jest ustawiona na DENY (anulować) lub REJECT (odrzuć), pakiet przechodzi dalej.

Demasquerade (demaskarada):

Jeśli pakiet jest odpowiedzią na poprzedni, zamaskarowany pakiet, jest demaskarowany i przechodzi bezpośrednio do łańcucha wyjściowego (output).

Routing decision (decyzja o rutingu):

Kod odpowiedzialny za ruting analizuje pole przeznaczenia, by zdecydować czy pakiet ma trafić do lokalnego procesu (sprawdź 'proces lokalny' poniżej) lub przekazany do zdalnej maszyny (sprawdź 'łańcuch przekazujący', również poniżej).

Local process (proces lokalny):

Proces działający na maszynie może otrzymywać pakiety po decyzji z poprzedniego punktu, jak również wysyłać pakiety (które po decyzji o rutingu z punktu wyżej, trafiają do łańcucha wyjściowego (output)).

lo interface (interfejs lo):

Jeśli pakiety z lokalnego procesu mają trafić do lokalnego procesu, przejdą przez łańcuch wyjściowy (output) z interfejsem ustawiony na 'lo', a potem wrócą przez łańcuch wejściowy (input), również przez interfejs 'lo' nazywany **pętlą zwrotną** (ang. loopback).

local (lokalnie):

Jeśli pakiet nie został wykreowany przez proces lokalny, sprawdzany jest łańcuch przekazujący (forward), w innym wypadku pakiet przekazywany jest do łańcucha wyjściowego (output).

forward chain (łańcuch przekazujący):

Łańcuch ten jest sprawdzany kiedy pakiet stara się przejść przez tą maszynę do innej.

output chain (łańcuch wyjściowy):

Ten łańcuch jest z kolei sprawdzany dla wszystkich pakietów, zanim opuszczą one maszynę.²⁷

Przykład:

```
ipchains -A input -d 192.168.1.2 110 -j accept
```

Na podstawie tej reguły akceptowane będą pakiety, jeżeli adres docelowy i port są prawidłowe. W powyższym przykładzie na port 110 będzie przyjmowany pakiet od lokalnego hosta o numerze IP 192.168.1.2 natomiast reguła:

```
ipchains -A input -p tcp -s 0/0 -d 212.160.100.10 139 -j  
DENY -i eth0
```

wskazuje, iż pakiety w protokole tcp pochodzące z każdego źródła i jakiegokolwiek portu skierowane na port nr 139 maszyny o IP 212.160.100.10 będą odrzucane i reguła ta dotyczy interfejsu eth0.

Wszystkie potrzebne reguły należy umieścić (np. */etc/rc.d/rc.firewall* lub w innym zależności od dystrybucji) możemy też sami stworzyć taki skrypt. W początkowej części takiego pliku należy umieścić linie:

```
ipchains -F input  
ipchains -F output  
ipchains -F forward  
ipchains -P input ACCEPT  
ipchains -P output ACCEPT  
ipchains -P forward DENY
```

A w następnej kolejności kierując się zasadą że co nie jest dozwolone jest zabronione należy zablokować wszystko (zarówno w filtrze *input*, jak i *output*):

²⁷ Linux IPCHAINS-HOWTO Bibliografia poz.12

```
ipchains -A input -s 0/0 -d 212.160.100.10 -j DENY -i eth0
ipchains -A output -s 0/0 -d 212.160.100.10 -j DENY -i eth0
```

Zakładając, iż adres 212.160.100.10 jest *IP* naszego serwera, to poprzez dodanie powyższej reguły zablokowaliśmy dostęp do naszego hosta, jak i wyjście przez interfejs eth0. Kolejnymi regułami w pliku powinny być reguły umożliwiające korzystanie przez określone hosty z określonych usług, np.:

```
ipchains -I input -p tcp -s 0/0 -d 212.160.100.10 22 -j
ACCEPT -i eth0
ipchains -I input -p tcp -s 0/0 -d 212.160.100.10 113 -j
ACCEPT -i eth0
```

Można tutaj zauważyć zastosowanie podobnych metod, jak przy kontroli dostępu w plikach **host.deny** i **host.allow**: najpierw zabronić wszystkim korzystania z jakichkolwiek usług, a następnie otwierać poszczególne porty umożliwiając tym samym na dostęp do serwera. Jeżeli natomiast chodzi o filtr **forward**, to właśnie w nim określamy reguły, dzięki którym sieć wewnętrzna może korzystać poprzez serwer z dostępu do sieci zewnętrznej. Dzieje się to dzięki opcji maskowania adresów. Jeżeli chcemy, aby nasz serwer stał się routerem udostępniającym poszczególnym komputerom w sieci wewnętrznej połączenie z siecią zewnętrzną, to wystarczy dodać np. taką regułę:

```
ipchains -A forward -s 192.168.1.2 -d 0/0 -j MASQ -i eth0
```

Zakładając, iż interfejs eth0 serwera jest wyjściem na świat (Internet), to zapis taki umożliwi korzystanie z zasobów globalnej sieci użytkownikowi komputera 192.168.1.2 z sieci lokalnej. Musimy pamiętać, że możliwość maskowania adresów jest również ściśle związana z opcjami, które powinny być zaznaczone podczas kompilacji samego jądra. Musimy też pamiętać aby przyznać prawa wykonania dla pliku /etc/rc.d/rc.firewall

```
$ chmod 700 /etc/rc.d/rc.firewall
```

Zmiany w tym pliku może dokonywać tylko root

6.2.2 Filtrowanie Protokołu ICMP

Przy filtrowaniu protokołu ICMP nie możemy odwołać się do danego portu, gdyż porty nie są numerowane. Dlatego odwołujemy się do typu i kodu

Typ liczbowy	Nazwa symboliczna	Opis
0	<i>echo -reply</i>	Odpowiedź ping
3	<i>destination unreachable</i>	Nieosiągalne miejsce przeznaczenia
4	<i>source-quench</i>	Kontrola przepływu między dwoma ruterami
5	<i>redirect</i>	Przekierowanie
8	<i>echo request</i>	Żądanie echa ping
11	<i>time exceeded</i>	Liczba ruterów przez, które przeszedł pakiet związana TTL
12	<i>parametr -problem</i>	W nagłówku IP nieoczekiwane wartości

Musimy pamiętać, że komunikaty ICMP służą do analizy sieci i często są używane w celu badania portów otwartych na naszym firewallu. Nie możemy jednak blokować wszystkich, gdyż mogą później wystąpić problemy z działaniem sieci.

Na pewno należy przepuścić pakiety typu 3

```
ipchains -A input -i eth0 -p icmp -s 0/0 3 -d  
212.160.100.10 -j ACCEPT
```

```
ipchains -A output -i eth0 -p icmp -s 0/0 3 -d
212.160.100.10 -j ACCEPT
```

W Linuksie istnieje możliwość odczytywania informacji, które są zawarte w nagłówku. Pozwalają nam one na lepsze filtrowanie pakietów przechodzących przez nasz firewall.

6.2.3 Filtrowanie na pakietów z ustawioną flagą

Czasami chcemy mieć możliwość połączenia *TCP* tylko w jedną stronę np. chcemy mieć możliwość nawiązywania połączenia z zewnętrznym serwerem WWW, ale żeby serwer nie mógł nawiązywać połączeń z nami. Najszybszym rozwiązaniem byłoby zablokowanie pakietów przychodzących od tego serwera. Niestety problem polega na tym, że w połączeniu *TCP* pakiety muszą krążyć w obydwie strony. Rozwiązaniem tego problemu jest blokada pakietów z prośbą o połączenie (pakiety z ustawioną flagą SYN) blokując te pakiety uniemożliwiamy nawiązanie próby połączeń. W tym celu używamy parametru ”-y” ale stosować go możemy tylko dla protokołu *TCP*.

6.2.4 Obsługa fragmentacji

Czasami pakiet jest zbyt duży dlatego jest on dzielony na fragmenty i wysyłany jako kilka pakietów. Nie jest zalecane przepuszczanie takich pakietów przez firewall. Zalecałbym ustawienie opcji w jądrze w taki sposób aby najpierw nastąpiło złożenie pakietu w całość przed przesłaniem go dalej. Unikniemy przez to kłopotów, gdyż zawsze możemy dostać pakiet spreparowany, który spowoduje zawieszenie się danej maszyny.

„Możemy zadać sobie pytanie co się dzieje po przejściu pakietu przez filtr:

1. Licznik dla reguły zliczającej bajty jest powiększany o rozmiar pakietu.
2. Licznik pakietów dla reguły jest zwiększany.

3. Jeżeli tak jest ustawione w regułach dany pakiet jest logowany.
4. Jeżeli reguła tego wymaga zmieniane jest pole Typu Usługi (*Type Of Service*).
5. Pakiet może być oznaczony.
6. Sprawdzany jest adres docelowy aby określić co zrobić dalej z pakietem.”²⁸

Jak wiemy reguły mogą przyjmować wartości: ACCEPT, DENY, REJECT, MASQ. Jednak dodatkowo można użyć opcji REDIRECT (przekierowanie), który mówi kernelowi aby wysłał pakiet do lokalnego portu zamiast tam gdzie miał się dostać. Możemy zastosować nazwę portu lub jego numer po opcji ”-j REDIRECT” pakiet zostanie przekierowany do tego portu, nawet jeśli był zaadresowany do innego. Ograniczeniem jest fakt, że możemy stosować tą politykę dla łańcuchów *input* i protokołów *TCP* i *UDP*. Istnieje możliwość definiowania własnych reguł są one opisywane małymi literami.

```
-s 192.168.0.1 -p ICMP -j REJECT
-s 192.168.0.1 -p TCP -j test -----> -s 192.168.0.1 -p 139 ACCEPT
-s 192.168.0.1 -p UDP -j DENY
```

Logowanie pakietów

Logowanie pakietów możemy uzyskać używając parametru ” l ” być może będziemy chcieli logować np. odwołania na port 139.

```
ipchains -A output -i eth0 -p all -s 0/0 -d 212.160.100.10
139 -j ACCEPT -l
```

²⁸ Linux IPCHAINS-HOWTO Bibliografia poz. 12

6.2.5 Zmiana Typ Usługi (ToS)

„Są to bity używane w nagłówku *IP* i powodują zmianę sposobu w jaki traktowany jest pakiet. Te cztery bity to „Minimum Delay” (Minimalna zwłoka), „Maximum Ththroughput” (Maksymalna przepustowość), „Maximum Reliability” (Maksymalna niezawodność) i „Minimum Cost” (Minimalny koszt). Zezwala się na ustawienie tylko jednego z tych bitów.

Najczęstszym ustawieniem jest danie połączeniom telnetowym i ftp atrybutu 'Minimalna Zwłoka' a danym ftp 'Maksymalna Przepustowość'. Może to wyglądać tak:

```
ipchains -A output -p tcp -d 0.0.0.0/0 telnet -t 0x01 0x10
ipchains -A output -p tcp -d 0.0.0.0/0 ftp -t 0x01 0x10
ipchains -A output -p tcp -s 0.0.0.0/0 ftp-data -t 0x01 0x08
```

Parametr '-t' przyjmuje dwa dodatkowe parametry, oba w zapisie heksdecymalnym. Pozwalają one na złożone manipulowanie bitami TOS jako maski: pierwsza służy do przeprowadzenia operacji logicznej AND na polu TOS aktualnego pakietu, a druga do operacji XOR na wyniku. Ilustruje to poniższa tabela:

Nazwa TOS	Wartość	Typowe zastosowanie
Minimum Delay	0x01 0x10	ftp, telnet
Maximum Throughput	0x01 0x08	ftp-data
Maximum Reliability	0x01 0x04	snmp
Minimum Cost	0x01 0x02	nntp

Nie zaleca się jednak zmiany tych wartości przy połączeniu modemowym.²⁹

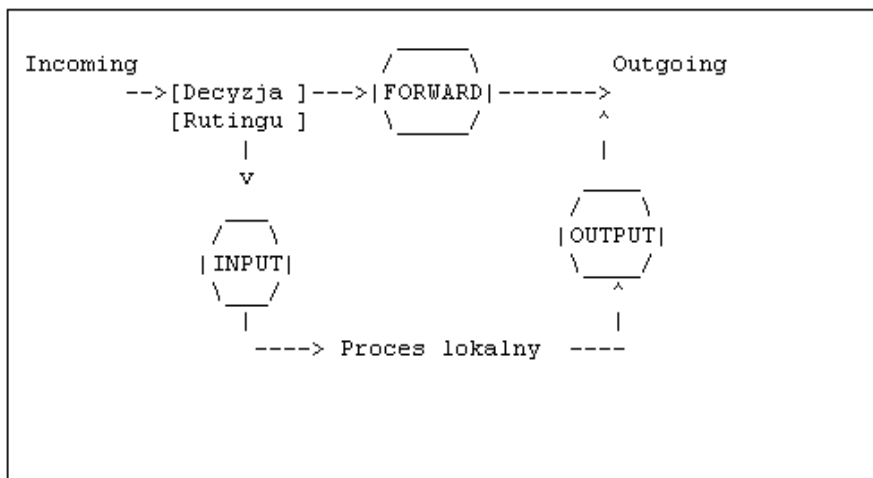
²⁹ Linux IPCHAINS-HOWTO Bibliografia poz. 12

Netfilter (iptables kernel 2.4)

Od jądra 2.3 ipchains zostało zamienione na iptables zmieniono trochę składnię oraz obsługę NAT. W celu śledzenia połączeń należy załadować moduł ip_conntrack. szczególnie gdy jest to firewall z NAT dodano dwa łańcuchy POSTROUTING i PREROUTING czyli przed i po routingu. Ma to na celu kierowanie pakietów, które nie muszą przechodzić przez wszystkie reguły ale mają być kierowane do danego miejsca np. wszystkie pakiety z numerem portu 80 kierowane będą do serwera WWW.

Sposób obsługi ruchu dla kernela 2.4

Rys. 21 Przebieg pakietów dla poszczególnych łańcuchów w netfilter.



Źródło: *Filtrowanie pakietów wLinuxie 2.4*

Łańcuchy INPUT, OUTPUT i FORWARD, pozostały tak jak w poprzedniej wersji .

Lista możliwych operacji na całym łańcuchach:

„Stworzenie nowego łańcucha (-N).

Skasowanie pustego łańcucha (-X).

Zmiana polityki dla wbudowanego łańcucha (-P).

Wylistowanie reguł w łańcuchu (-L).

Wyczyszczenie łańcucha z reguł (-F).

Wyzerowanie liczników pakietów i bajtów we wszystkich regułach w łańcuchu (-Z).

Zmiana reguł w obrębie łańcucha:

Dodanie nowej reguły do łańcucha (-A).

Wstawienie nowej reguły na pewnej pozycji w łańcuchu (-I).

Zamiana reguły na pewnej pozycji w łańcuchu (-R).

Skasowanie reguły na pewnej pozycji w łańcuchu (-D).

Skasowanie pierwszej reguły pasującej do podanej w łańcuchu (-D).

Rozszerzenia TCP

Rozszerzenia TCP ładowane są automatycznie gdy podano opcję '-p tcp'. Dodają one następujące opcje :

--tcp-flags

Po której następuje opcjonalny znak '!', a następnie dwa ciągi flag, które pozwalają wskazać zestaw flag do zbadania. Drugi ciąg mówi, które mają być ustawione. Na przykład:

```
# iptables -A INPUT --protocol tcp --tcp-flags ALL SYN,ACK  
-j DROP
```

mówi, że sprawdzone powinny zostać wszystkie flagi ('ALL' to synonim dla 'SYN,ACK,FIN,RST,URG,PSH'), ale tylko flagi SYN i ACK powinny być ustawione. Istnieje również argument 'NONE' który oznacza, że żadna flaga nie może być ustawiona.

--syn

opcjonalnie poprzedzona przez '!', jest skrótem dla '--tcp-flags SYN,RST,ACK SYN'.

--source-port

po której następuje opcjonalny '!', a następnie pojedynczy port lub grupa portów TCP. Porty mogą być wskazywane przez nazwy takie jak w /etc/services lub przez numery. Grupy portów wskazuje się albo przez dwie nazwy portów podzielone przez ':' lub wskazując zakres portów 22:300

--sport

to synonim '--source-port'.

--destination-port

oraz

--dport

mają takie same opcje jak powyżej, ale określają port przeznaczenia

--tcp-option

po którym następuje opcjonalny znak ! i numer, które wskazują na opcję TCP równą wskazanemu numerowi. Pakiet który nie posiada kompletnego nagłówka TCP jest automatycznie odrzucany jeśli wykonana zostanie próba sprawdzenia jego opcji TCP.³⁰

6.2.6 Rozszerzenia UDP

Są one ładowane automatycznie po podaniu '-p udp'. Udostępniają opcję '--source-port', '--sport', '--destination-port' i '--dport', dokładnie takie same jak dla TCP powyżej.

³⁰ Filtrowanie pakietów w Linuksie 2.4 Bibliografia poz. 12

6.2.7 Rozszerzenia ICMP

Ładowane automatycznie po podaniu '-p icmp'. Udostępniają one tylko jedną nową opcję:

--icmp-type

po której następuje opcjonalny znak '!', a następnie albo nazwa typu pakietu ICMP (np. 'host-unreachable', czyli komputer nieosiągalny), lub numer typu (np. '3'), lub numer typu i kod oddzielone przez '/' (np. '3/3'). Lista dostępnych nazw typów ICMP dostępna jest po podaniu '-p icmp --help'.

6.2.8 Test Stanu

„Najbardziej użytecznym testem jest ten dostarczany przez rozszerzenie 'state', który interpretuje analizę śledzenia połączeń modułu 'ip_contrack'. Podanie w regule opcji '-m state' udostępnia dodatkową opcję '--state', która jest listą oddzielonych stanów do przetestowania (opcja '!' wskazuje na pakiety nie pasujące do wskazanych stanów).

Stanami, które można sprawdzać są:

NEW (NOWY)

Pakiet, który tworzy nowe połączenie.

ESTABLISHED (NAWIĄZANY)

Pakiet, który należy do istniejącego połączenia (np. pakiet odpowiedzi, lub pakiet wychodzący w połączeniu które otrzymało już odpowiedzi).

RELATED (POWIĄZANY)

Pakiet, który jest powiązany z istniejącym połączeniem, ale nie jest jego częścią, tj. np. pakiet z błędem ICMP, lub (jeśli załadowany jest moduł FTP) pakiet ustanawiający połączenie ftp dla danych.

INVALID (BŁĘDNY)

Pakiet, który nie może być zidentyfikowany z jakiś powodów: mogą to być wyczerpanie się pamięci, lub błędy ICMP, które nie należą do żadnego znanego połączenia. Generalnie, pakiety tego typu powinno się odrzucać.

Różnice pomiędzy iptables i ipchains

- Nazwy wbudowanych łańcuchów zostały zmienione z małych na DUŻE litery, ponieważ łańcuchy INPUT i OUTPUT otrzymują tylko pakiety kierowane do maszyny lokalnej lub pakiety generowane lokalnie. Do tej pory używano ich dla wszystkich pakietów przychodzących i wychodzących.
- Opcja '-i' oznacza teraz interfejs wejściowy i działa tylko w łańcuchach INPUT i FORWARD. Reguły w łańcuchach FORWARD i OUTPUT zamiast dotychczasowego '-i' używają '-o'.
- Porty TCP i UDP podaje się teraz przez opcje --source-port lub --sport (czy też --destination-port/--dport) i zapisuje po wskazaniu protokołu ('-p tcp' lub '-p udp') ponieważ dopiero one ładują rozszerzenia odpowiednio TCP i UDP
- Opcję TCP '-y' zmieniono na '--syn', i należy ją zapisać po '-p tcp'.
- Cel DENY nazywa się DROP (WYRZUCIĆ).
- Działa zerowanie pojedynczych łańcuchów z jednoczesnym wylistowaniem ich zawartości.
- Działa zerowanie wbudowanych łańcuchów łącznie z kasowaniem liczników wywołania polityki dla danego łańcucha
- Listowanie łańcuchów dostarcza ci „zdjęcia” (ang. snapshot) liczników.
- REJECT i LOG są teraz celami rozszerzonymi, co oznacza że są osobnymi modułami kernela.
- Nazwy łańcuchów mogą mieć do 31 znaków.
- Opcja MASQ nazywa się MASQUERADE i używa innej składni. REDIRECT zachował nazwę, ale również zmieniono składnię.

- Opcja '-o' nie jest już używana by kierować pakiety do urządzenia z przestrzeni użytkownika (należy sprawdzić opcję '-i'). Wysyła się je tam poprzez skierowanie do celu QUEUE.³¹

6.3 NAT w Linuksie

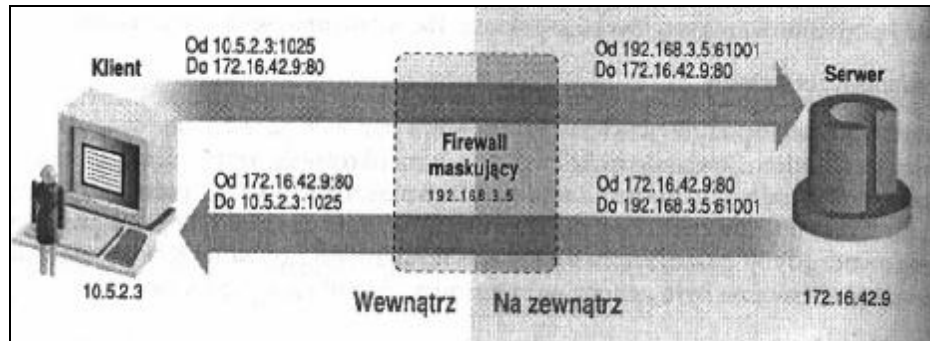
Maskarada w Linuksie potrafi pracować na poziomie wyższych protokołów i potrafi przeprowadzić głębsze zmiany niż proste podmienianie adresów, znana jest jako przezroczysty system pośredniczący (*transparent proxy system*). Trudno tu wykazać czy jest to pośredniczenie czy filtrowanie, jest to coś pośredniego między jednym a drugim. Jak wiemy adres *IP* jest używany do komunikacji ze zdalnymi usługami. W przypadku prostych protokołów maskarada zmienia tylko informacje w nagłówku *IP*, włącznie z adresami *IP*, numerami portów, numerami sekwencyjnymi i numerami potwierżeń. Maskarada używa adresu *IP* hosta przeprowadzającego maskaradę jako adresu widzianego zewnątrz, i mapuje port na jeden z puli 4096 numerów poczynając od 61000. Ta stała lokacja ogranicza maskaradę do 4096 połączeń *TCP* i 4096 portów *UDP*. Z reguły Linuks przydziela porty o numerach poniżej 32768 tak więc nie kolidują one z innymi usługami. Maskarada może współpracować z takimi protokołami (*FTP*, *RealAudio*), które nie zbyt dobrze współpracują z *NAT*. Osiągalne to jest dzięki ładownym dynamicznym modułom.

6.3.1 Jak działa NAT w Linuksie

„Maskarada przechwytuje pakiety przesyłane przez jądro Linuksa. W pakietach wychodzących modyfikowane są adresy *IP* i numery portów. W przypadku *TCP* generowany jest nowy numer sekwencyjny. Proces ten jest odwrócony w przypadku pakietów przychodzących.

³¹ Filtrowanie pakietów w Linuksie 2.4 Bibliografia poz. 12

Rys. 20 Maskarada dla połączeń wychodzących.

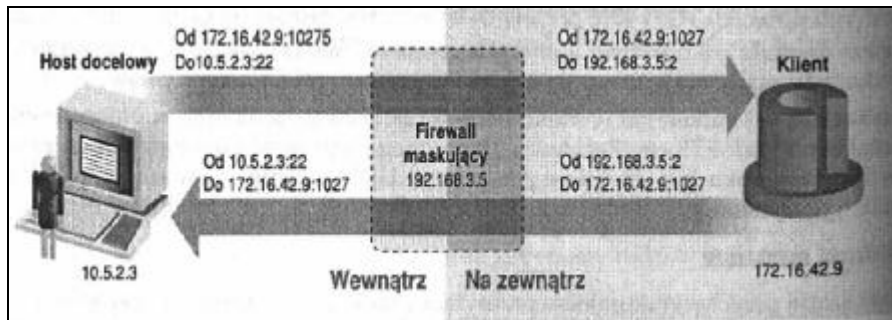


Źródło: *Internet Firewalls* wyd. RM str.194

Powyżej mamy przedstawioną przykładową sesję ze zdalnym serwerem HTTP. Firewall maskujący będzie przekazywał pakiety do klienta, dopóki klient będzie miał otwarte połączenie *TCP*. W przypadku *UDP* firewall będzie przekazywał pakiety do klienta tylko przez konfigurowalny czas, który przeważnie wynosi od 15-30 sekund. Oprócz obsługi połączeń wychodzących maskarada może obsługiwać przekazywanie połączeń przychodzących do usług wewnętrznych. Możliwość maskowania portów wejściowych jest konfigurowana statycznie dla każdego portu, który ma być przekazywany. Gdy port jest przekazywany, nie może być używany do łączenia się z usługą w firewallu. Jako przykład możemy użyć sesji SSH do wewnętrznego miejsca przeznaczenia i pokazywane są adresy *IP* i numery portów każdej ze strony połączenia. Możliwe jest przekazywanie tego samego portu do kilku innych celów, gdy firewall maskujący jest skonfigurowany tak aby nasłuchiwał na wielu adresach *IP*.³²

³² Internet firewalls wyd RM 193-194

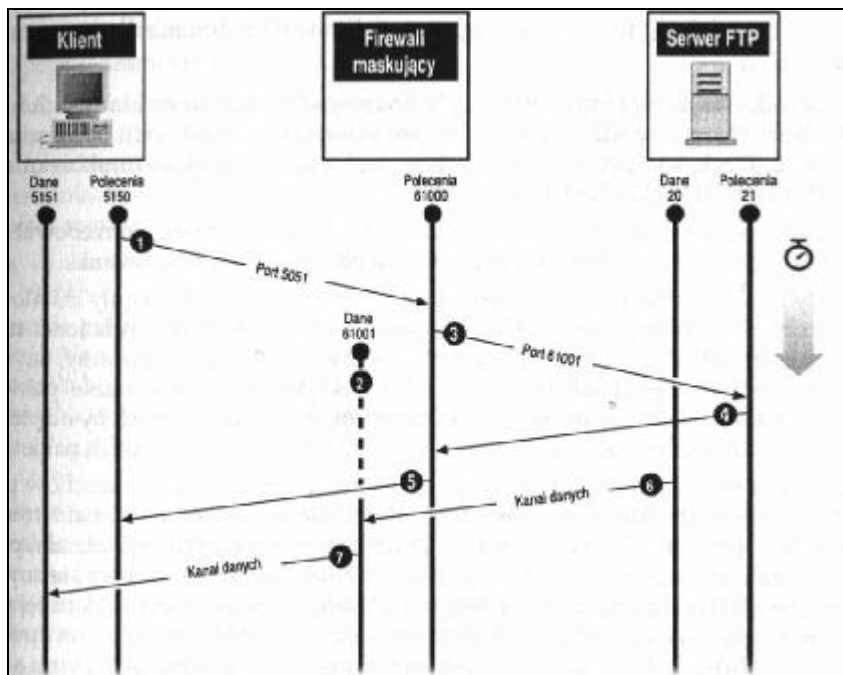
Rys 21 NAT dla połączeń przychodzących.



Źródło: *Internet Firewalls* wyd. RM str.194

W wypadku bardziej skomplikowanych protokołów maskarada może nasłuchiwać w dodatkowych portach *TCP* i *UDP*, w zależności od zawartych widzianych pakietów. Może nawet przepisać od nowa zawartość pakietu, aby zamienić adresy *IP* i numery portów. Najlepszym przykładem jest sesja *FTP* problem polega na tym, że *FTP* używa połączeń od serwera do klienta.

Rys 22 NAT dla połączenia *FTP* w trybie normalnym.



Źródło: *Internet Firewalls* wyd. RM str.195

Klient FTP otwiera kanał kontrolny do pożądanego serwera FTP. W miejscu gdzie ma się rozpocząć transfer danych, klient wydaje polecenie PORT, które zawiera adres *IP* klienta i numer portu, na którym klient będzie czekał na dane. Serwer FTP używa tych informacji do otwarcia nowego połączenia TCP do klienta w celu przesłania danych. Aby maskarada mogła działać, musi przejąć polecenie PORT, wydawane przez klienta. Moduł maskarady FTP robi to podsłuchując wszystkie polecenia przesyłane przez wszystkie kanały kontrolne FTP. Gdy zobaczy polecenie PORT, najpierw ustanawia na hoście maskaradującym tymczasowy port, który będzie przekazywany na port podany przez klienta. Następnie przepisuje pakiet *IP*, zawierający polecenie PORT, używając adresu firewala i portu tymczasowego. Gdy zostanie nawiązane połączenie z tymczasowym portem, będzie on przekazywany do klienta.”³³

Polecenia ipchains używane w NAT:

```
-M - L    aktualne zmaskarowane połączenia
-M - S    parametry maskaradowania
```

Do parametru ”-S” powinny zawierać tzw. parametry timeout’ów, każdy w sekundach: dla TCP, dla sesji TCP po pakiecie FIN i dla pakietów UDP. Domyślną wartością są 15 min, 2 min, 5min. (/usr/src/linux/include/net/ip_masq.h)

Dodatkowo w celu obsługi nietypowych usług, które chcemy udostępnić istnieje możliwość ładowania dodatkowych modułów np:

```
insmod ip_masq_ftp
insmod ip_masq_irc
insmod ip_masq_quake
insmod ip_masq_raudio
insmod ip_masq_user
insmod ip_masq_vdolive
```

³³ Internet Firewalls wyd. RM str. 194-195

```
insmod ip_masq_cuseeme
insmod ip_masq_portfw
```

6.3.1 NAT dla jądra 2.4

W jądrze 2.4 przepisano całą obsługę NAT, zmieniając ją dodano łańcuch POSTROUTING i PREROUTING. Istnieje możliwość przekazywania portów (*ang. port-forwarding*) a co za tym idzie rozkładania obciążenia (*ang. load-sharing*) co powoduje, że można rozłożyć np. zapytania do serwera WWW na inne maszyny na, których dostępna jest dana usługa.

W kernelu 2.4 istnieją dwa typy NAT źródłowy SNAT i docelowy DNAT;

SNAT ma miejsce gdy zmieniamy adres źródłowy pierwszego pakietu tzn. zmienimy adres maszyny, z której inicjowane jest połączenie. Musimy pamiętać, że SNAT wykonywany jest zawsze po routingu (*ang. post-routing*), tuż przed tym jak pakiet opuści maszynę.

DNAT ma miejsce gdy zmieniamy adres docelowy pierwszego pakietu tzn. zmieniamy adres maszyny do, której ma dotrzeć połączenie. DNAT wykonywany jest zawsze przed routingiem (*ang. pre-routing*), kiedy pakiet właśnie został odebrany. Przekazywanie portów, rozkładanie obciążenia i transparentne proxy to różne rodzaje DNAT.

Aby w ogóle zainicjować NAT musimy załadować moduł, który jest odpowiedzialny za jego obsługę. Jeżeli wkompilewaliśmy go w jądro nie musimy tego robić; ładowanie modułu jądra:

```
modprobe iptable_nat /musimy pamiętać, że usunięte zostaną inne/
```

Dodatkowo należy włączyć przekazywanie pakietów IP

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Należy jeszcze włączyć maskarowanie pakietów

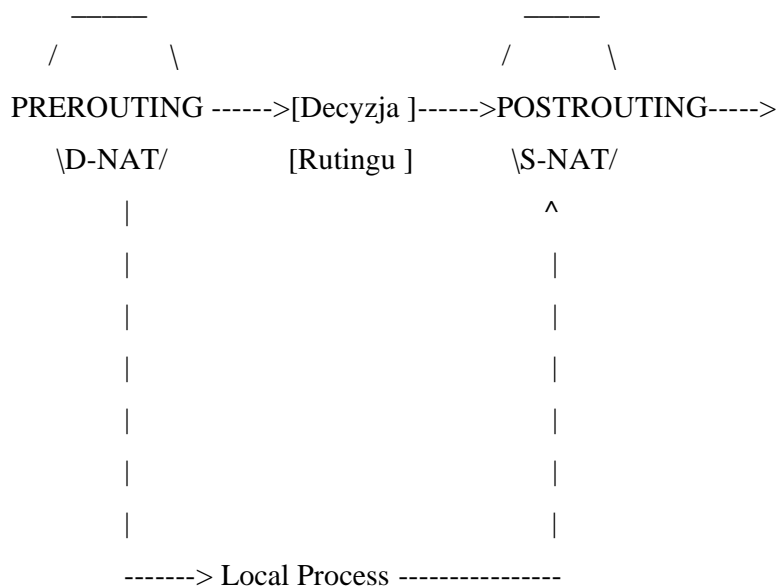
```
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

Kernel nie wie jakie połączenia ma modyfikować musimy zatem stworzyć reguły NAT. Możemy je dopisać w tym samym skrypcie, który odpowiada za działanie firewalla. Służy do tego wywołanie NAT w iptables

```
iptables -t nat
```

Tabela NAT zawiera dwa łańcuchy każda reguła jest sprawdzana dopóki się nie okaże, że któraś nie pasuje. Łańcuchy te nazywają się: PREROUTING (dla DNAT, ponieważ pakiety najpierw do nas docierają), POSTROUTING (dla SNAT, po którym pakiety opuszczają nasz system) i OUTPUT (dla DNAT, dla pakietów generowanych lokalnie). Przykład obsługi NAT dla kernela 2.4.

Rys. 23 Obsługa NAT dla jądra od wersji 2.4 (Linuks)



Źródło: *Linux_2_4 NAT HOWTO*

W każdym z tych punktów, gdy dociera pakiet sprawdzana jest reguła z jakim połączeniem jest skojarzony. Jeżeli jest to nowe połączenie sprawdzany jest łańcuch w tabeli NAT jeżeli taki wpis istnieje dotyczyć on będzie wszystkich przyszłych pakietów dla tego połączenia.

Składnia poleceń :

```
iptables -t nat / Wybiera tabele w tym wypadku będzie to NAT/  
-A / dodaje regułę na koniec łańcucha POSTROUTING /  
-I / dodaje regułę na początek łańcucha PREROUTING /
```

Możemy podawać adresy źródłowe (-s lub - - source) a także i docelowe (-d lub - - destinations) jako adres IP możemy podawać adres pojedynczy (192.168.1.5) nazwę (www.gumis.pl) jak i też sieć 192.168.0.0/24.

Możemy wskazywać też interfejs, którego ma dotyczyć wejściowy (-i lub - - in-interface) lub wyjściowy (-o lub - - out - interface). Jednak musimy pamiętać o tym, że wybór interfejsu jest podyktowany faktem reguł POSTROUTING dla interfejsu wyjściowego a PREROUTING dla interfejsu wejściowego. Należy również wskazywać protokół (-p lub - protocol). Możemy zatem przyporządkować dany protokół co umożliwi nam zmianę reguł pamiętając jednak, że związane są z tym dodatkowe opcje -source-port i -destination-port. Pozwalają nam one określić o jakie dokładnie pakiety nam chodzi i które porty będą wykorzystywane. Dodatkowo musimy użyć opcji (-p). Opcję tę możemy wykorzystać jeżeli mamy w sieci wewnętrznej serwer WWW, który musi być dostępny z zewnątrz. Możemy dokładnie określić, że wywołania na port 80 będą przekierowane do serwera WWW na port 8080. W tym wypadku wykorzystywane jest DNAT i wykonywany jest PREROUTING zaraz przed tym jak pakiet zostanie odebrany, co spowoduje, że jeżeli pakiet zostanie później poddany routingowi lub filtrowi pakietów będzie już skierowany do adresu docelowego np. WWW możemy w tym wypadku używać opcji -i dla interfejsu wejściowego (możemy również użyć przekierowania REDIRECT)

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -i  
eth0 -j DNAT --to 192.168.0.8:8080
```

W takim wypadku wszystkie pakiety przychodzące na port 80 będą poddane DNAT i skierowane do maszyny 192.168.0.8:8080 na port 8080.

Dodatkowo NAT wykonuje mapowanie portów. Działanie to polega na tym, że jeżeli np. klient nawiązuje połączenie na porcie 1024 a jeżeli jest zajęty następuje przemapowanie portu, na port o innym numerze żeby umożliwić klientowi nawiązane połączenia.

6.3.2 DHCP a firewall

Problemem z jakim możemy się spotkać jest brak stałego adresu *IP* ale przyznawany jest on z serwera DHCP providera. Nowy klient do obsługi DHCP to pump. Problem polega na tym, że najpierw musimy odczytać nasz adres IP a później przekazać go do skryptu, który startuje firewalla. Musimy też pamiętać że najpierw musi nastąpić pobranie adresu zaś później wpisanie go do reguł w przeciwnym wypadku nasz firewall nie będzie działał. W tym celu należy utworzyć skrypt powłoki w katalogu */etc/sysconfig/network-scripts/ifdhcp-done* podstawowe zadanie tego skryptu to informowanie skryptu */sbin/ifup* czy *dhcpd* czy zostały pobrane ustawienia konfiguracyjne z serwera DHCP. Skrypt od firewalla musi odczytać zmienne *IPAADR* i *NAMESERVER* z pliku */etc/dhcp/dhcpd-eth0.info*³⁴

³⁴ Pełny przykład skryptu „Linux Firewalls” wyd Robomatic str. 416

VII. Proxy

Z pośród serwerów proxy możemy rozróżnić :

7.1 Serwery proxy działające na poziomie aplikacji i na poziomie obwodu.

Serwery takie zwane też aplikacyjnymi, opierają się na znajomości aplikacji, dla której pośredniczy tzn. rozumie i interpretuje polecenia w protokole aplikacji. Serwery działające na poziomie obwodu opierają się na tworzeniu obwodu między klientami i serwerem bez interpretowania protokołu aplikacji. Jako przykład serwera działającego na poziomie aplikacji możemy podać Sendmaila, który posługuje się protokołem typu „przechowaj i prześlij dalej”. Zaś typowe proxy obwodowe to programy typu (*plung-gw*), które przejmują wszystkie otrzymane dane i przesyłają je do innego miejsca przeznaczenia. Zaletą serwerów obwodowych jest to, że mogą one świadczyć usługi dla wielu różnych protokołów. Wadą zaś jest to, że nie zapewniają kontroli nad przekazywanymi danymi

7.2 Inteligentne serwery proxy.

Serwery proxy mogą znacznie więcej niż tylko przekazywać żądania. Serwer, który wykonuje dodatkowe operacje np: serwer proxy HTTP, który buforuje dane, uwierzytelnia użytkowników, szyfruje sesję itp.

7.3 SOCKS

Obecnie używane są dwie wersje SOCKS: SOCKS 4 i SOCKS 5. Nie są one ze sobą zgodne ale większość serwerów SOCKS 5 wykrywa próby użycia SOCKS 4 i potrafi odpowiednio zareagować. Główne dodatki w SOCKS 5 to:

- uwierzytelnianie użytkowników
- UDP i ICMP

- Odwzorowywanie nazwy hostów w serwerze SOCKS

SOCKS 4 nie uwierzytelnia użytkowników. Podejmuje decyzję o dopuszczeniu lub odrzuceniu połączenia na podstawie tych samych informacji, których używają filtry pakietów (porty źródłowe o docelowe adresy IP). SOCKS 5 zapewnia obsługę kilku metod uwierzytelniania użytkowników, co umożliwia bardziej precyzyjną kontrolę i rejestrowanie zdarzeń. Dodatkowo wadą SOCKS 4 jest fakt, że działa on tylko z *TCP*; nie działa z *UDP* czy też z *ICMP*. W takim wypadku jesteśmy zmuszeni użyć SOCKS 5 lub programu UDP Packet Relayer. Program świadczy podobne usługi dla klientów *UDP* co SOCKS dla *TCP*. Należy tu wspomnieć, że SOCKS 5 to jedyny powszechnie używany, bezpłatny serwer proxy dla *ICMP*. SOCKS 4 wymaga, aby klient mógł tłumaczyć nazwy hostów na adresy *IP*. W SOCKS 5 klient może dostarczyć nazwę hosta zamiast adresu *IP*, w takim wypadku serwer SOCKS sam zajmie się odwzorowaniem nazwy hosta na dany adres. Przydatne jest to zwłaszcza w sieciach, które używają DNS z fałszywym serwerem głównym (*fake root*), gdzie hosty wewnętrzne używają czysto wewnętrznego serwera nazw, nie porozumiewającego się z Internetem. Cechą SOCKS jest to, że jest on bardzo uniwersalny a co za tym idzie ma ograniczoną funkcjonalność. Między innymi nie kontroluje on ani nie rejestruje zdarzeń specyficznych dla danych protokołów. SOCKS może rejestrować żądania połączenia z serwerem, umożliwia kontrolę dostępu według użytkownika, adresu i portu źródłowego oraz docelowego a także pozwala na skonfigurowanie odpowiedzi w przypadku odmowy dostępu np. informuje administratora o próbach połączenia z zewnątrz i powiadamia użytkowników dlaczego odmówiono im dostępu. W przypadku używania SOCKS musimy używać programów, które współpracują z SOCKS tzn. „uSOCKSyfikowanych” klientów takich jak np. Telnet, FTP.

7.4 TIS (Internet Firewall Toolkit)

W przeciwieństwie do pakietu SOCKS, który udostępnia pojedynczy serwer proxy, TIS dostarcza serwer proxy dla każdej usługi związanej z usługą internetową. Użycie małych a zarazem oddzielnych programów ze wspólnym plikiem konfiguracyjnym ma zapewnić bezpieczeństwo wynikające z zastosowania inteligentnych serwerów umożliwiając jednocześnie centralne sterowanie.

FTP

TIS może pośredniczyć dla FTP za pomocą specjalnych programów klienckich, lub też za pomocą specjalnych procedur użytkownika (*ftp-gw*). Jeżeli chcemy udostępnić serwer FTP musimy skorzystać z specjalnych procedur użytkownika. I to właśnie jest najczęściej spotykana konfiguracja FTP w TIS. Problem polega na tym, że nie ma klientów, z którymi mógłby współpracować TIS. Ponieważ jest to wyspecjalizowany serwer proxy pozwala on na rejestrowanie, odrzucanie i dodatkowe uwierzytelnianie poszczególnych poleceń FTP.

Telnet

Serwery TIS dla telnetu (*telnet-gw*) rlogin (*rlogin-gw*) obsługują specjalne procedury użytkownika. Użytkownicy mogą łączyć się z serwerem jednak tak naprawdę zgłoszenie logowania otrzymują od serwera proxy.

Uniwersalne proxy TIS

TIS zawiera uniwersalne proxy (*plug-gw*), jednak nie obsługuje on dużej ilości protokołów co ogranicza jego zastosowanie. TIS bada tylko adres, z którego nadeszło połączenie i port z którego przyszło a następnie nawiązuje połączenie z innym hostem przez inny port. Wiemy jednak jak można prosto te zabezpieczenia obejść.

HTTP

TIS potrafi obsługiwać też HTTP i Gophera za pomocą programu (*http-gw*). Jednak obsługuje on tylko specjalne procedury użytkownika lub specjalnie do tego przystosowanych klientów. Większość klientów HTTP może skorzystać z pośredniczenia ale trzeba wskazać mu gdzie znajduje się serwer proxy.

7.5 SQUID

Jest to serwer proxy zaliczany do serwerów cache (buforowanie) ma on za zadanie przechowywanie danych na dysku. Klient łączy się z serwerem SQUID, który zamiast klienta kieruje zapytania do innych serwerów.

Serwer Squid oferuje następujące możliwości:

- proxy i buforowanie dla HTTP i FTP
- proxy dla SSL
- hierarchiczne buforowanie
- zarządzanie SNMP
- obsługa protokołów ICP (*Intrenet Cache Protocol*), HTCP, CARP

Najważniejszą cechą Sqiuda jest to, że najczęściej używane obiekty są składowane w pamięci RAM a nie na dysku co znacznie przyśpiesza działanie tego serwera.

VIII. Załączniki

*Przykładowe pliki konfiguracyjne*³⁵

Załącznik A

I. Sieć LAN z jednym adresem IP z NAT.

Sieć będzie udostępniała połączenie z Internetem

```
#!/bin/sh
# /etc/rc.d/rc.firewall
# eth0 = 213.168.12.10 /Interfejs na styku LAN/INTERNET
ppp0
# eth1= 192.168.0.1 /Interfejs dla sieci wewnętrznej LAN
# Możemy utworzyć alias
alias ipchains ='/sbin/ipchains/

# Włączamy przekazywanie pakietów
echo 1 > /proc/sys/net/ipv4/ip_forward

# Ochrona przed fałszowaniem pakietów
echo 1 > /proc/sys/net/ipv4/tcp_syncookies

# Weryfikacja adresów źródłowych
echo 1 > /proc/sys/net/ipv4/conf/all/rp_filter;do
echo 1>$f
done

# Ładowanie modułów do obsługi maskarady
insmod ip_masq_ftp
insmod ip_masq_irc
insmod ip_masq_quake
insmod ip_masq_raudio
insmod ip_masq_user
insmod ip_masq_vdolive
```

³⁵ Skrypty napisano przy pomocy dokumentacji Bibliografia pozy. 12 i 13

```

insmod ip_masq_cuseeme
insmod ip_masq_portfw

# Czyszczenie reguł firewalla
ipchains -F input
ipchains -F output
ipchains -F forward
# Polityka domyślnego zakazu
ipchains -F input          DENY
ipchains -F output REJECT
ipchains -F forward REJECT

# Możemy konfigurować firewalla postępując według polityki:
# 1.Blokujemy wszystko i uruchamiamy usługi, które są nam
# potrzebne
# 2.Uruchamiamy usługi a resztę blokujemy

# Wszystkie połączenia wychodzące mają być maskaradowane
ipchains -A forward -s 192.168.0.0/24 -j MASQ

# Dostęp dla pętli zwrotnej looback
ipchains -A input -i 127.0.0.1 -j ACCEPT
ipchains -A output -i 127.0.0.1 -j ACCEPT

# Blokujemy adresy pochodzące z interfejsu pętli zwrotnej
ipchains -A input -i ppp0 -s 127.0.0.1 -j DENY -l
ipchains -A input -i ppp0 -s 127.0.0.1 -j DENY -l

# Dostęp do serwera DNS
ipchains -A output -i ppp0 -p udp -s 192.168.0.0/24
1024:65535 -d 0/0 53 -j ACCEPT
ipchains -A input -i ppp0 -p udp -s 0/0 53
-d 192.168.0.0/24 1024:65535 -j ACCEPT

```

```

# Dostęp do serwerów WWW
ipchains -A output -i ppp0 -p tcp -s 192.168.0.0/24
-d 0/0 www -j ACCEPT

ipchains -A input -i ppp0 -p tcp !-y -s 0/0 www -d
192.168.0.0/24 -j ACCEPT

# Dostęp do serwerów WWW SSL
ipchains -A output -i ppp0 -p tcp -s 192.168.0.0/24 -d 0/0
443 -j ACCEPT
ipchains -A input -i ppp0 -p tcp !-y -s 0/0 443 -d
192.168.0.0/24 1024:65535 -j ACCEPT

# E-mail SMTP
ipchains -A output -i ppp0 -p tcp -s 192.168.0.0/24
-d 0/0 25 -j ACCEPT
ipchains -A input -i ppp0 -p tcp !-y -s 0/0 25
-d 192.168.0.0/24 -j ACCEPT

# POP
ipchains -A output -i ppp0 -p tcp -s 192.168.0.0/24
-d 0/0 110 -j ACCEPT
ipchains -A input -i ppp0 -p tcp !-y -s 0/0 110
-d 192.168.0.0/24 -j ACCEPT

# IMAP
ipchains -A output -i ppp0 -p tcp -s 192.168.0.0/24
-d 0/0 143 -j ACCEPT
ipchains -A input -i ppp0 -p tcp !-y -s 0/0 143
-d 192.168.0.0/24 -j ACCEPT

# Serwery grup dyskusyjnych NNTP 119
ipchains -A output -i ppp0 -p tcp -s 192.168.0.0/24

```

```

-d 0/0 119 -j ACCEPT

ipchains -A input -i ppp0 -p tcp !-y -s 0/0 119
-d 192.168.0.0/24 -j ACCEPT

# Telnet 23 /klient/
ipchains -A output -i ppp0 -p tcp -s 192.168.0.1 23
-d 0/0 -j ACCEPT
ipchains -A input -i ppp0 -p tcp !-y -s 0/0 23
-d 192.168.0.0/24 1024:65535 -j ACCEPT

#SSH
ipchains -A output -i ppp0 -p tcp -s 192.168.0.0/24
1024:65535 -d 0/0 ssh -j ACCEPT
ipchains -A input -i ppp0 -p tcp !-y -s 0/0 ssh
-d 192.168.0.0/24 1024:65535 -j ACCEPT

#FTP porty 21,22
# Dostęp do zdalnych serwerów FTP /klient/
ipchains -A output -i ppp0 -p tcp -s 192.168.0.0/24 -d 0/0
21 -j ACCEPT
ipchains -A input -i ppp0 -p tcp !-y -s 0/0 21 -d
192.168.0.0 1024:65535 -j ACCEPT

# FTP kanał danych tryb zwykły
ipchains -A input -i ppp0 -p tcp -s 0/0 20 -d
192.168.0.0/24 1024:65535 -j ACCEPT
ipchains -A output -i ppp0 -p tcp !-y -s 192.168.0.0/24
1024:65535 -d 0/0 20 -j ACCEPT

# Kanał FTP tryb pasywny /przeglądarki WWW/
ipchains -A output -i ppp0 -p tcp -s 192.168.0.0/24
1024:65535 -d 0/0 1024:65535 -j ACCEPT

```

```
ipchains -A input -i ppp0 -p tcp !-y -s 0/0 1024:65535 -d
192.168.0.0/24 1024:65535 -j ACCEPT
```

```
# Blokujemy pozostałą resztę
```

```
ipchains -A input -i ppp0 -p all -s 0/0 1:65535 -d
192.168.0.0/24 -j DENY
```

```
ipchains -A output -i ppp0 -p all -s 192.168.0.0 -d 0/0
1:65535 -j DENY
```

II. Sieć LAN z NAT (kernel 2.4)

```
#!/bin/sh
```

```
#Musimy włączyć przekazywanie pakietów
echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
# Ochrona przed fałszowaniem pakietów
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

```
# Weryfikacja adresów źródłowych
echo 1 > /proc/sys/net/ipv4/conf/all/rp_filter;do
echo 1>$f
done
```

```
# Ładowanie modułów kernela
```

```
insmod ip_tables
```

```
insmod ip_contrack
```

```
insmod ip_contrack_ftp
```

```
# Moduł ip_contract służy do śledzenia połączeń
```

```
# Czyścimy reguły
```

```
iptables -F
```

```
iptables -F nat
```

```
iptables -F mangle
```

```
# Blokujemy całego firewalla
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT DROP

# Dopuszczamy loopback
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# Dopuszczamy realizacje połączeń dla poszczególnych
# protokołów
iptables -A FORWARD -p all -j ACCEPT -m state --state
ESTABLISHED
iptables -A FORWARD -p all -j ACCEPT -m state --state
RELATED

# Pozwalamy na realizacje połączeń z adresów
iptables -A FORWARD -s 192.168.0.0/24 -m --state NEW -j
ACCEPT

# Maskarada na wyjściu
iptables -t nat -A POSTROUTING -p all 192.168.0.0/24 -j
MASQUERADE

# Możemy rozszerzyć ten skrypt wzorując się na skrypcie dla
ipchains używając składni iptables
```

III. Sieć LAN z DMZ

Sieć będzie udostępniała podstawowe usługi np. WWW, Poczta, DNS. Zamiast serwerów WWW,FTP, Mail skieruje zapytania klientów do serwerów proxy w DMZ.

```
#!/bin/sh
# /etc/rc.d/rc.firewall

INTERNET="ppp0"           /Interfejs do Internetu
DMZ_INTER="eth1"         /Interfejs dla sieci DMZ
LAN2_INTER="eth2"        /Interfejs do sieci LAN
LAN3_INTER="eth3"        /Interfejs na styku LAN/DMZ
PROXY="192.168.0.1"       /Serwer Proxy
LAN1="192.168.0.0/28"     /Adresy w sieci DMZ
LAN2="10.0.0.0/16"       /Adresy w sieci wewnętrznej
ANY="0/0"                /Dowolny adres
LOOPBACK="127.0.0.0/8"   /Pętla zwrotna
# Możemy utworzyć alias
alias ipchains ='/sbin/ipchains/

#Musimy włączyć przekazywanie pakietów
echo 1 > /proc/sys/net/ipv4/ip_forward

# Ochrona przed fałszowaniem pakietów
echo 1 > /proc/sys/net/ipv4/tcp_syncookies

# Weryfikacja adresów źródłowych
echo 1 > /proc/sys/net/ipv4/conf/all/rp_filter;do
echo 1>$f
done
```

```

# Ładowanie modułów do obsługi maskarady
insmod ip_masq_ftp
insmod ip_masq_irc
insmod ip_masq_quake
insmod ip_masq_raudio
insmod ip_masq_user
insmod ip_masq_vdolive
insmod ip_masq_cuseeme
insmod ip_masq_portfw

# Czyszczenie reguł firewalle
ipchains -F input
ipchains -F output
ipchains -F forward

# Polityka domyślnego zakazu
ipchains -F input DENY
ipchains -F output REJECT
ipchains -F forward REJECT

# Dostęp dla pętli zwrotnej looback
ipchains -A input -i $LOOBACK -j ACCEPT
ipchains -A output -i $LOOBACK -j ACCEPT

# Musimy teraz skonfigurować dwa firewalle jeden będzie na
# granicy DMZ/INTERNET a drugi na styku DMZ/LAN który
# pełni rolę zapory dławiącej

# Firewall DMZ/INTERNET

# Wszystkie połączenia wychodzące mają być maskaradowane
ipchains -A forward -s $LAN1 -d $ANY -j MASQ

```

```

# Blokujemy adresy pochodzące z interfejsu pętli zwrotnej
ipchains -A input -i $INTERNET -s $LOOPBACK -j DENY -l
ipchains -A input -i $INTERNET -s $LOOPBACK -j DENY -l

# Dostęp do serwera DNS
ipchains -A output -i $INTERNET -p udp -s $PROXY
1024:65535 -d $ANY 53 -j ACCEPT
ipchains -A input -i $INTERNET -p udp -s $ANY 53
-d $PROXY 1024:65535 -j ACCEPT

# Dostęp do serwerów WWW
ipchains -A output -i $INTERNET -p tcp -s $PROXY
-d $ANY www -j ACCEPT
ipchains -A input -i $INTERNET -p tcp !-y -s $ANY www -d
$PROXY -j ACCEPT

# Dostęp do serwerów WWW SSL
ipchains -A output -i $INTERNET -p tcp -s $PROXY -d $ANY
443 -j ACCEPT
ipchains -A input -i $INTERNET -p tcp !-y -s $ANY 443 -d
$PROXY 1024:65535 -j ACCEPT

# E-mail SMTP
ipchains -A output -i ppp0 -p tcp -s $PROXY
-d $ANY 25 -j ACCEPT
ipchains -A input -i ppp0 -p tcp !-y -s $ANY 25
-d $PROXY -j ACCEPT

# POP
ipchains -A output -i ppp0 -p tcp -s $PROXY
-d $ANY 110 -j ACCEPT
ipchains -A input -i ppp0 -p tcp !-y -s $ANY 110

```

```

-d $PROXY -j ACCEPT

# IMAP
ipchains -A output -i ppp0 -p tcp -s $PROXY
-d $ANY 143 -j ACCEPT
ipchains -A input -i ppp0 -p tcp !-y -s $ANY 143
-d $PROXY -j ACCEPT

#SSH
ipchains -A output -i ppp0 -p tcp -s $PROXY
1024:65535 -d $ANY ssh -j ACCEPT

ipchains -A input -i ppp0 -p tcp !-y -s $ANY ssh
-d $PROXY 1024:65535 -j ACCEPT

#FTP porty 21,22
# Dostęp do zdalnych serwerów FTP /klient/
ipchains -A output -i ppp0 -p tcp -s $PROXY
-d $ANY 21 -j ACCEPT
ipchains -A input -i ppp0 -p tcp !-y -s $ANY 21 -d
$PROXY 1024:65535 -j ACCEPT

# FTP kanał danych tryb zwykły
ipchains -A input -i ppp0 -p tcp -s $ANY 20 -d
$PROXY 1024:65535 -j ACCEPT
ipchains -A output -i ppp0 -p tcp !-y -s $PROXY
1024:65535 -d $ANY 20 -j ACCEPT

# Kanał FTP tryb pasywny /przeglądarki WWW/
ipchains -A output -i ppp0 -p tcp -s $PROXY
1024:65535 -d $ANY 1024:65535 -j ACCEPT
ipchains -A input -i ppp0 -p tcp !-y -s $ANY 1024:65535 -d
$PROXY 1024:65535 -j ACCEPT

```

```

# Blokujemy pozostała resztę
ipchains -A input -i ppp0 -p all -s $ANY 1:65535 -d
$PROXY -j DENY
ipchains -A output -i ppp0 -p all -s $PROXY
-d $ANY 1:65535 -j DENY

# Firewall DMZ/LAN
# Reguły będą podobne jak wyżej ale skierowane będą do
# serwera Proxy
# Sieć wewnętrzna
INTERNET="ppp0" /Interfejs do Internetu/DMZ
DMZ_INTER="eth1" /Interfejs dla sieci DMZ
LAN2_INTER="eth2" /Interfejs do sieci LAN
LAN3_INTER="eth3" /Interfejs na styku LAN/DMZ
PROXY="192.168.0.1" /Serwer Proxy
LAN1="192.168.0.0/28" /Adresy w sieci DMZ
LAN2="10.0.0.0/16" /Adresy w sieci wewnętrznej
ANY="0/0" /Dowolny adres
LOOPBACK="127.0.0.0/8" /Pętla zwrotna

# Dostęp dla pętli zwrotnej loopback
ipchains -A input -i $LOOPBACK -j ACCEPT
ipchains -A output -i $LOOPBACK -j ACCEPT

# Blokujemy adresy pochodzące z interfejsu pętli zwrotnej
ipchains -A input -s $LOOPBACK -j DENY -l
ipchains -A input -s $LOOPBACK -j DENY -l

# Dostęp do serwera DNS
ipchains -A output -i $LAN3_INTER -p udp -s $PROXY
1024:65535 -d $LAN2 53 -j ACCEPT
ipchains -A input -i $LAN2_INTER -p udp -s $LAN2 53
-d $PROXY 1024:65535 -j ACCEPT

```

```

# Dostęp do serwerów WWW
ipchains -A output -i $LAN3_INTER -p tcp -s $PROXY
-d $LAN2 www -j ACCEPT

ipchains -A input -i $LAN2_INTER -p tcp !-y -s $LAN2 www
-d $PROXY -j ACCEPT

# Dostęp do serwerów WWW SSL
ipchains -A output -i $LAN3_INTER -p tcp -s $PROXY
-d $LAN2 443 -j ACCEPT
ipchains -A input -i $LAN2_INTER -p tcp !-y -s $LAN2 443
-d $PROXY 1024:65535 -j ACCEPT

# E-mail SMTP
ipchains -A output -i $LAN3_INTER -p tcp -s $PROXY
-d $LAN2 25 -j ACCEPT
ipchains -A input -i $LAN2_INTER -p tcp !-y -s $LAN2 25
-d $PROXY -j ACCEPT

# POP
ipchains -A output -i $LAN3_INTER -p tcp -s $PROXY
-d $LAN2 110 -j ACCEPT
ipchains -A input -i $LAN2_INTER -p tcp !-y -s $LAN2 110
-d $PROXY -j ACCEPT

# IMAP
ipchains -A output -i $LAN3_INTER -p tcp -s $PROXY
-d $LAN2 143 -j ACCEPT
ipchains -A input -i $LAN2_INTER -p tcp !-y -s $LAN2 143
-d $PROXY -j ACCEPT

#SSH

```

```

ipchains -A output -i $LAN3_INTER -p tcp -s $PROXY
1024:65535 -d $LAN2 ssh -j ACCEPT
ipchains -A input -i $LAN2_INTER -p tcp !-y -s $LAN2 ssh
-d $PROXY 1024:65535 -j ACCEPT

#FTP porty 21,22
# Dostęp do zdalnych serwerów FTP /klient/
ipchains -A output -i $LAN3_INTER -p tcp -s $PROXY
-d $LAN2 21 -j ACCEPT
ipchains -A input -i $LAN2_INTER -p tcp !-y -s $LAN2_INTER
21 -d $PROXY 1024:65535 -j ACCEPT

# FTP kanał danych tryb zwykły
ipchains -A input -i $LAN2_INTER -p tcp -s $LAN2 20
-d $PROXY 1024:65535 -j ACCEPT
ipchains -A output -i $LAN3_INTER -p tcp !-y -s $PROXY
1024:65535 -d $LAN2 20 -j ACCEPT

# Kanał FTP tryb pasywny /przeglądarki WWW/
ipchains -A output -i $LAN3_INTER -p tcp -s $PROXY
1024:65535 -d $LAN2 1024:65535 -j ACCEPT
ipchains -A input -i $LAN2_INTER -p tcp !-y -s $LAN2
1024:65535 -d $PROXY 1024:65535 -j ACCEPT

# Blokujemy pozostałą resztę
ipchains -A input -i ppp0 -p all -s $ANY 1:65535 -d
$PROXY -j DENY
ipchains -A output -i ppp0 -p all -s $PROXY
-d $ANY 1:65535 -j DENY

```

Załącznik B

W tym miejscu musimy zastanowić się nad faktem w jaki sposób będziemy mogli stwierdzić, że ktoś próbował dokonać włamania do naszego systemu. Należy zatem analizować logi w tym celu należy zmienić ustawienia `sysloga`, który jest odpowiedzialny za logowanie pakietów oczywiście należy pamiętać o wskazaniu w regułach firewalla, które pakiety chcemy logować. Musimy zmienić ustawienia w pliku `/etc/syslog.conf` w którym dokonamy następującego wpisu

```
kern.* /var/log/firewall
```

Analizując ten plik możemy spotkać się z takimi wpisami (oczywiście plik ten jest o wiele większy dlatego dostosowałem go w celu przedstawienia problemu).

Zaczynając od prawej strony :

1. Kiedy pakiet został zalogowany data nazwa systemu
2. IN –na jakim interfejsie
3. OUT –oznacza, z którym łańcuchem firewalla mamy do czynienia
4. Adresy MAC
5. SRC – adres nadawcy
6. DST – adres odbiorcy
7. LEN –długość pakietu
8. TOS –Type of Service
9. TTL – Czas życia
10. ID - Identyfikator datagramu, pakietu
11. PROTO –protokół
12. SPT – port źródłowy nadawcy
13. DPT – port docelowy (Telnet)
14. WINDOW – wielkość okna
15. SYN URGP – flagi ustawione w nagłówku TCP

W tym przypadku został zalogowany pakiet sesji Telnet, która wcześniej została zablokowana w regułach ściany ogniowej. Pozostałe przedstawione logi stanowią próby nawiązania sesji pocztowej 110, SMB 136-139 oraz sesji WWW 8080. Pozostałe logi w dokumentacji dołączonej na płycie CD – R.

Logi z firewalla

Jun 7 14:26:33 labD-05 kernel: IN=eth0 OUT=
MAC=00:e0:7d:a1:50:b7:00:e0:7d:a1:4e:9a:08:00 SRC=10.0.1.224
DST=10.0.1.225 LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=30 DF PROTO=TCP
SPT=1027 DPT=23 WINDOW=16384 RES=0x00 SYN URGP=0

Jun 7 14:26:36 labD-05 kernel: IN=eth0 OUT=
MAC=00:e0:7d:a1:50:b7:00:e0:7d:a1:4e:9a:08:00 SRC=10.0.1.224
DST=10.0.1.225 LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=31 DF PROTO=TCP
SPT=1027 DPT=23 WINDOW=16384 RES=0x00 SYN URGP=0

Jun 7 14:26:42 labD-05 kernel: IN=eth0 OUT=
MAC=00:e0:7d:a1:50:b7:00:e0:7d:a1:4e:9a:08:00 SRC=10.0.1.224
DST=10.0.1.225 LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=32 DF PROTO=TCP
SPT=1027 DPT=23 WINDOW=16384 RES=0x00 SYN URGP=0

Jun 7 14:28:22 labD-05 kernel: IN=eth0 OUT=
MAC=00:e0:7d:a1:50:b7:00:e0:7d:a1:4e:9a:08:00 SRC=10.0.1.224
DST=10.0.1.225 LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=52 DF PROTO=TCP
SPT=1029 DPT=110 WINDOW=16384 RES=0x00 SYN URGP=0

Jun 7 14:28:24 labD-05 kernel: IN=eth0 OUT=
MAC=00:e0:7d:a1:50:b7:00:e0:7d:a1:4e:9a:08:00 SRC=10.0.1.224
DST=10.0.1.225 LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=53 DF PROTO=TCP
SPT=1029 DPT=110 WINDOW=16384 RES=0x00 SYN URGP=0

Jun 7 14:28:30 labD-05 kernel: IN=eth0 OUT=
MAC=00:e0:7d:a1:50:b7:00:e0:7d:a1:4e:9a:08:00 SRC=10.0.1.224
DST=10.0.1.225 LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=54 DF PROTO=TCP
SPT=1029 DPT=110 WINDOW=16384 RES=0x00 SYN URGP=0

Jun 7 14:41:39 labD-05 kernel: IN=eth0 OUT=
MAC=00:e0:7d:a1:50:b7:00:e0:7d:a1:4e:9a:08:00 SRC=10.0.1.224
DST=10.0.1.225 LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=60 DF PROTO=TCP
SPT=1031 DPT=110 WINDOW=16384 RES=0x00 SYN URGP=0

Jun 7 14:41:45 labD-05 kernel: IN=eth0 OUT=
MAC=00:e0:7d:a1:50:b7:00:e0:7d:a1:4e:9a:08:00 SRC=10.0.1.224
DST=10.0.1.225 LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=61 DF PROTO=TCP
SPT=1031 DPT=110 WINDOW=16384 RES=0x00 SYN URGP=0

Jun 7 14:44:30 labD-05 kernel: IN=eth0 OUT=
MAC=ff:ff:ff:ff:ff:ff:00:e0:7d:a1:4e:98:08:00 SRC=10.0.1.101
DST=10.0.1.255 LEN=96 TOS=0x00 PREC=0x00 TTL=128 ID=225 PROTO=UDP
SPT=137 DPT=137 LEN=76

Jun 7 14:44:31 labD-05 kernel: IN=eth0 OUT=
MAC=ff:ff:ff:ff:ff:ff:00:e0:7d:a1:4e:98:08:00 SRC=10.0.1.101
DST=10.0.1.255 LEN=96 TOS=0x00 PREC=0x00 TTL=128 ID=318 PROTO=UDP
SPT=137 DPT=137 LEN=76

Jun 7 14:44:31 labD-05 kernel: IN=eth0 OUT=
MAC=ff:ff:ff:ff:ff:ff:00:e0:7d:a1:4e:9a:08:00 SRC=10.0.1.224
DST=10.0.1.255 LEN=229 TOS=0x00 PREC=0x00 TTL=128 ID=65 PROTO=UDP
SPT=138 DPT=138 LEN=209

Jun 7 14:44:32 labD-05 kernel: IN=eth0 OUT=
MAC=ff:ff:ff:ff:ff:ff:SRC=10.0.1.101 DST=10.0.1.255 LEN=259 TOS=0x00
PREC=0x00 TTL=128 ID=396 PROTO=UDP SPT=138 DPT=138 LEN=239

Jun 7 14:44:40 labD-05 kernel: IN=eth0 OUT=
MAC=ff:ff:ff:ff:ff:ff:00:e0:7d:a1:4e:98:08:00 SRC=10.0.1.101
DST=10.0.1.255 LEN=216 TOS=0x00 PREC=0x00 TTL=128 ID=398 PROTO=UDP
SPT=138 DPT=138 LEN=196

Jun 7 14:44:41 labD-05 kernel: IN=eth0 OUT=
MAC=ff:ff:ff:ff:ff:ff:00:e0:7d:a1:4e:98:08:00 SRC=10.0.1.101
DST=10.0.1.255 LEN=96 TOS=0x00 PREC=0x00 TTL=128 ID=399 PROTO=UDP
SPT=137 DPT=137 LEN=76

Jun 7 14:44:41 labD-05 kernel: IN=eth0 OUT=
MAC=ff:ff:ff:ff:ff:ff:00:e0:7d:a1:4e:98:08:00 SRC=10.0.1.101
DST=10.0.1.255 LEN=216 TOS=0x00 PREC=0x00 TTL=128 ID=403 PROTO=UDP
SPT=138 DPT=138 LEN=196

Jun 7 14:44:42 labD-05 kernel: IN=eth0 OUT=
MAC=ff:ff:ff:ff:ff:ff:00:e0:7d:a1:4e:98:08:00 SRC=10.0.1.101
DST=10.0.1.255 LEN=216 TOS=0x00 PREC=0x00 TTL=128 ID=476 PROTO=UDP
SPT=138 DPT=138 LEN=196

Jun 7 14:44:PREC=0x00 TTL=128 ID=69 DF PROTO=TCP SPT=1034 DPT=8080
WINDOW=16384 RES=0x00 SYN URGP=0

Jun 7 14:47:34 labD-05 kernel: IN=eth0 OUT=
MAC=00:e0:7d:a1:50:b7:00:e0:7d:a1:4e:9a:08:00 SRC=10.0.1.224
DST=10.0.1.225 LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=70 DF PROTO=TCP
SPT=1034 DPT=8080 WINDOW=16384 RES=0x00 SYN URGP=0

Jun 7 14:47:40 labD-05 kernel: IN=eth0 OUT=
MAC=00:e0:7d:a1:50:b7:00:e0:7d:a1:4e:9a:08:00 SRC=10.0.1.224
DST=10.0.1.225 LEN=48 TOS=0x00 PREC=0x00 TTL=128 ID=71 DF PROTO=TCP
SPT=1034 DPT=8080 WINDOW=16384 RES=0x00 SYN URGP=0

Wnioski

Analizując możliwości budowy firewalla opartego o systemy uniksowe możemy zauważyć, że jest to bardzo elastyczna platforma a ilość dostępnych narzędzi może budzić zdziwienie. Z reguły jest tak że do ochrony sieci wystarczy tylko jeden program np. Microsoft Proxy Serwer . W systemach uniksowych możemy zauważyć, że wybór jest dość szeroki: od serwerów proxy poczynając na typowych firewallach kończąc. Jeżeli do tego dodamy możliwość realizacji firewalli stanowych opartych na badaniu stanu połączenia daje nam to możliwość zbudowania naprawdę dobrego urządzenia zabezpieczającego naszą sieć a najważniejsze, że koszty są bardzo małe w porównaniu do typowych rozwiązań sprzętowych takich firm jak Cisco, Checkpoint. Jednak nie spodziewajmy się dużej wydajności przy przesyłaniu dużej ilości danych. Jak wiemy zawsze rozwiązania sprzętowe są lepsze od programowych, jednak te drugie mają bardzo dużą przewagę ponieważ są elastyczne pozwalają na szybką modyfikację i zmiany konfiguracji w stosunku do danej chwili. Nie musimy zatem czekać na nowy produkt tylko wymieniamy oprogramowanie. Jednego czego nie realizują firewalle zbudowane o systemy uniksowe typu: ipchains, iptables, IP-Filter to tego, że nie mają implementacji dla protokołu IPX/SPX. Jednak czy to jest aż tak ważne skoro nawet firma NetWare wycofuje się z tego sztandarowego protokołu w sieciach Novell i przechodzi na TCP/IP. Pisząc tą pracę trudno mi powiedzieć, które rozwiązanie jest najlepsze i zanim zaczniemy budować firewalla musimy się głęboko zastanowić co chcemy tak naprawdę chronić. Przeważnie rozwiązania takie dostosowuje się do sieci jaką mamy. Coraz lepsze rozwiązania oprogramowania odpowiedzialnego za budowę ścian ogniowych wskazują na to, że skuteczność ochrony będzie coraz większa. Rozwiązań jest w tym wypadku bardzo dużo i nie będę tu porównywał, co jest lepsze czy proxy czy firewall, podobnie jak nie rozstrzygniemy, co jest mądrzejsze jajko czy kura. Każdy użytkownik dostosuje sobie takie rozwiązanie jakie jest mu najbardziej potrzebne a systemy uniksowe w pełni mu to umożliwią.

Zakończenie

W mojej pracy starałem się zebrać i przeanalizować nie samą tylko budowę firewalla, ale także całą ideę budowy ścian ogniowych. Pomimo, że temat tej pracy wspomina tylko o systemach uniksowych, nie możemy zapomnieć o tym, że ściany ogniowe wspierają działanie całych sieci korporacyjnych, które muszą działać bezawaryjnie i nie sposób zrealizować tego tematu nie analizując tego, jakie są trendy i rozwiązania na świecie. Starałem się w mojej pracy bardziej zgłębić temat firewalli niż tylko omówić budowę samych reguł, jakie muszą być ustawione, aby zabezpieczać daną sieć. Myślę, że po części mi się to udało.

Praca ta dała mi bardzo dużo satysfakcji umożliwiając zapoznanie się z bardzo obszernym tematem, jakim jest ochrona sieci komputerowych.

Oświadczenie

Oświadczam, że pracę niniejszą przygotowałem/ am samodzielnie.

Wszystkie dane, istotne myśli i sformułowania pochodzące z literatury (przytoczone dosłownie lub niedosłownie) są opatrzone odpowiednimi odsyłaczami. Praca ta nie była, w całości ani części, która by zawierała znaczące fragmenty przedstawione w pracy jako oryginalne (wyniki badań empirycznych, obliczenia, spostrzeżenia, oceny, wnioski, propozycje itp.), przez nikogo przedłożona do żadnej oceny i nie była dotychczas publikowana.

Bibliografia :

1. Internet Firewalls Elizabeth D. Zwicky, Simon Cooper, D.Brent Chapman
wyd. RM Warszawa 2001.
1. Nie tylko wirusy Andrzej Dudek wyd. Helion
2. Hakerzy cała prawda Joel Scambray, Stuart McClure, George Kurtz
wyd. Translator s.c Warszawa 2001
3. Hakerzy w Linuksie Jamews Lee Brian Hatch George Kurtz wyd. Translator s.c
Warszawa 2002
4. Firewalls ściany ogniowe Matthew Strebe, Charles Perkins wyd. Mikom
Warszawa 2000
5. Linux Firewalls Robert L. Ziegler wyd. Robomatic 2001 Wrocław
6. Serwery sieciowe Linuksa Craig Hunt wyd. Mikom Warszawa 2000
7. Sieci komputerowe TCP/IP Douglas E. Comer wyd. WNT 1998 Warszawa
8. Chip Special Linux (Zima 2001)
9. PC Kurier (dział Archiwum)
10. Linux + (Styczeń 2002)
11. Win.Security Magzaine
12. www.mrOvka.eu.org/tlumaczenia
13. www.linux.centromost.com.pl
14. www.jtz.org.pl
15. www.linux.labs.eu.org