



**POLITECHNIKA WROCŁAWSKA**  
**WYDZIAŁOWY ZAKŁAD INFORMATYKI**  
**WYDZIAŁ INFORMATYKI I ZARZĄDZANIA**  
**Wybrzeże Wyspiańskiego 27, 50-370 Wrocław**

**PRACA MAGISTERSKA**

**OBRONA PRZED ATAKAMI TYPU ODMOWA USŁUGI (DoS)**

**Marcin Żurakowski**  
mzurakowski@esolution.pl

**Promotor:** dr inż. Przemysław Kazienko

**Ocena:**

---

**Wrocław 2004**

<b>1. Wstęp .....</b>	<b>5</b>
<b>2. Protokoły TCP/IP .....</b>	<b>6</b>
2.1. Protokół IP .....	7
2.2. Protokół ICMP .....	10
2.3. Protokół UDP .....	10
2.4. Protokół TCP .....	11
2.4.1. Nawiązywanie połączenia .....	13
<b>3. Ataki typu DoS/DDoS .....</b>	<b>15</b>
3.1. Fazy ataku .....	16
3.2. Historia ataków DoS/DDoS .....	18
<b>4. Podział i klasyfikacja ataków typu DDoS/DoS .....</b>	<b>20</b>
4.1. Klasyfikacja na podstawie stopnia automatyzacji .....	20
4.1.1. Ataki ręczne .....	20
4.1.2. Półautomatyczne ataki .....	20
4.1.2.1. Ataki z bezpośrednią komunikacją .....	20
4.1.2.2. Ataki z pośrednią komunikacją .....	21
4.1.3. Automatyczne ataki .....	21
4.1.3.1. Strategie skanowania .....	21
4.2. Propagacja kodu .....	22
4.3. Klasyfikacja na podstawie wykorzystywanych słabości ofiary .....	23
4.3.1. Ataki na protokół .....	23
4.3.2. Ataki siłowe .....	23
4.3.2.1. Ataki możliwe do wyfiltrowania .....	24
4.3.2.2. Ataki nie możliwe do wyfiltrowania .....	24
4.4. Klasyfikacja ze względu na dynamikę ataku .....	24
4.4.1. Ataki ciągłe .....	24
4.4.2. Ataki zmienne .....	25
4.4.2.1. Ataki ze wzrastającą siłą .....	25
4.4.2.2. Ataki z zmienną siłą .....	25
4.5. Klasyfikacja ze względu na skutek ataku DoS/DDoS .....	25
4.5.1. Ataki blokujące .....	25
4.5.2. Ataki obniżające poziom .....	25
<b>5. Podział i klasyfikacja metod obrony .....</b>	<b>27</b>
5.1. Mechanizmy prewencyjne .....	27
5.1.1. Zapobieganie atakom .....	27
5.1.1.1. Zmiany w bezpieczeństwie systemów .....	27
5.1.1.2. Zmiany w bezpieczeństwie protokołów .....	28
5.1.2. Zapobieganie odmowie usługi .....	28
5.1.2.1. Limitowanie zasobów .....	29
5.1.2.2. Mnożenie zasobów .....	29
5.2. Mechanizmy reaktywne .....	29
5.2.1. Wykrywanie ataków .....	29
5.2.1.1. Mechanizmy wykrywania ataków na podstawie wzorca .....	29
5.2.1.2. Wykrywanie ataków na podstawie anomalii .....	30
5.2.1.3. Hybrydowe wykrywanie ataków .....	30

5.2.1.4.	Wykrywanie ataków na podstawie zewnętrznych sygnałów .....	30
5.2.2.	Reakcja na atak.....	31
5.2.2.1.	Identyfikacja agentów agresora.....	31
5.2.2.2.	Limitowanie ruchu .....	31
5.2.2.3.	Filtrowanie ruchu .....	31
5.2.2.4.	Rekonfiguracja .....	31
5.2.3.	Mechanizmy autonomiczne.....	32
5.2.4.	Mechanizmy kooperatywne .....	32
5.2.5.	Mechanizmy współzależne .....	32
5.2.5.1.	Mechanizmy obrony w sieci ofiary .....	32
5.2.5.2.	Mechanizmy obrony w sieci pośredniej.....	32
5.2.5.3.	Mechanizmy obrony w sieci agresora.....	32
<b>6.</b>	<b>Popularne ataki typu DoS i DDoS .....</b>	<b>34</b>
6.1.	Powódź pakietów SYN - SYN Flood.....	34
6.1.1.	Anatomia ataku .....	34
6.1.2.	Przykłady ataków .....	35
6.1.3.	Obrona.....	35
6.1.3.1.	Ogólne metody obrony.....	35
6.1.3.2.	Linux .....	36
6.1.3.3.	Windows 2000/XP/2003 .....	37
6.2.	Atak typu LAND.....	40
6.2.1.	Anatomia ataku .....	40
6.2.2.	Obrona.....	41
6.3.	Ataki z wykorzystaniem powielaczy.....	41
6.3.1.	Wykorzystanie protokołu ICMP .....	42
6.3.1.1.	Atak typu SMURF .....	43
6.3.1.1.1.	Anatomia ataku .....	43
6.3.1.1.2.	Obrona.....	44
6.3.1.2.	ICMP – Redirect.....	45
6.3.1.2.1.	Anatomia ataku .....	46
6.3.1.2.2.	Obrona.....	47
6.3.2.	UDP.....	47
6.3.2.1.	DNS.....	48
6.3.2.1.1.	Anatomia ataku .....	48
6.3.2.1.2.	Obrona.....	49
6.3.2.2.	Chargen .....	49
6.3.2.2.1.	Anatomia ataku .....	50
6.3.2.2.2.	Obrona.....	50
6.3.3.	TCP.....	51
6.3.3.1.	HTTP .....	51
6.3.3.1.1.	Anatomia ataku .....	51
6.3.3.1.2.	Obrona.....	52
6.4.	Ping of Death.....	52
6.4.1.	Anatomia ataku .....	52
6.4.2.	Obrona.....	53
6.5.	Teardrop .....	53
6.5.1.	Anatomia ataku .....	53
6.5.2.	Obrona.....	54

<b>7. Narzędzia atakującego .....</b>	<b>55</b>
7.1. Trinoo .....	55
7.2. TFN (Tribe Flood Network) .....	57
7.3. Stacheldraht .....	57
<b>8. Projekt i implementacja systemu wykrywania ataków DOS/DDOS – BIDS .....</b>	<b>58</b>
8.1. Koncepcja działania systemu .....	59
8.2. Opis użytkowy .....	61
8.2.1. Miejsce instalacji systemu .....	61
8.2.2. Wymagania i instalacja aplikacji .....	64
8.2.3. Parametry systemu BIDS .....	65
8.2.4. Obserwacja parametrów sieci – dane wejściowe dla konfiguracji systemu .....	67
8.2.5. Działanie systemu .....	69
8.3. Opis techniczny .....	70
8.3.1. Moduły wewnętrzne .....	70
8.3.1.1. Moduł Catcher .....	71
8.3.1.2. Moduł Storman .....	71
8.3.1.3. Moduł Analyser .....	72
8.3.1.4. Moduł Analyser – Bayes .....	73
8.3.1.4.1. Funkcja dyskretyzacji .....	76
8.3.1.4.2. Uczenie sieci Bayesa .....	79
8.3.2. Formaty plików .....	82
8.3.2.1. Plik net_stat.conf .....	82
8.3.2.2. Plik config.h .....	83
8.3.2.3. Plik net.stat .....	85
8.3.2.4. Plik CASE .....	86
8.4. Analiza działania systemu BIDS .....	86
8.4.1. Środowisko testowe .....	86
8.4.2. Testy skuteczności .....	88
8.4.3. Testy wydajnościowe .....	95
<b>9. Podsumowanie .....</b>	<b>101</b>
<b>10. Bibliografia .....</b>	<b>102</b>
<b>11. Spisy .....</b>	<b>106</b>
11.1. Spis tabel .....	106
11.2. Spis rysunków .....	106
<b>12. Dodatki .....</b>	<b>108</b>
12.1. Przykładowe dane – statystyka sieci .....	108
12.2. Przykładowe dane – plik CASE .....	110
12.3. Struktury wykorzystywane w systemie .....	114
12.4. Funkcje biblioteki PCAP .....	116

# 1. Wstęp

Powstające w latach 70'tych sieci ARPANET, NFSNET dały początek Internetowi, czyli powszechnie znanej i wykorzystywanej globalnej pajęczynie. W zamysłach projektantów sieć miała służyć przede wszystkim celom wojskowym lub naukowym: zapewniać łączność, wspomagać przeprowadzanie badań i wymiany informacji pomiędzy ośrodkami. Zapewne żaden z jej twórców nie spodziewał się, że kiedyś Internet będzie siecią opłatającą cały świat i jednym z ważniejszych filarów światowej gospodarki. Dziś Internet łączy miliony ludzi a w jego wirtualnej rzeczywistości możemy znaleźć coraz to więcej usług pochodzących ze świata rzeczywistego (e-handel, e-gospodarka, e-wybory, e-rozrywka).

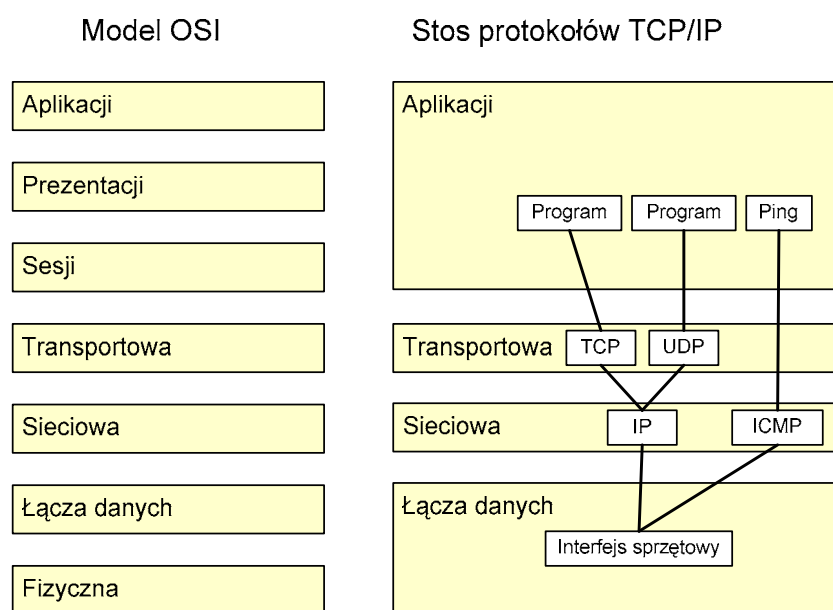
Internet ma też swoją ciemną stronę: włamania do systemów, blokowanie usług, wirusy, spam to tylko kilka zagrożeń czyhających na ludzi używających Internetu. Błędy popełnione we wstępnym latach podczas projektowania protokołów i usług wraz z niedoskonałym i dziurawym oprogramowaniem, spowodowały, że te niedoskonałości zamieniły się w prawdziwe plagi.

Jednym z takich zagrożeń są ataki typu DoS/DDoS (Denial of Service/Distributed Denial of Service), czyli działania mające na celu zablokowanie wybranej usługi i uniemożliwienie korzystania z niej uprawnionym użytkownikom. Celem niniejszej pracy jest omówienie tego rodzaju ataków. Koncentruję się ona na podgrupie tych ataków występujących w warstwie sieciowej i transportowej modelu OSI [1].

Praca podzielona jest na dwie części. W pierwszej przedstawione zostaną rozważania na temat ataków typu DoS/DDoS rozpoczynając od krótkiej charakterystyki rodziny protokołów TCP/IP, wprowadzenia do tego rodzaju ataków oraz historii. W dalszych rozdziałach zostanie dokonana klasyfikacja ataków oraz metod obrony oraz dokładny opis kilku popularnych ataków tego typu. Część teoretyczna kończy się na krótkiej charakterystyce kilku narzędzi agresora. Natomiast druga część pracy to praktyczna implementacja systemu BIDS (Bayesian Intrusion Detection System) służącego do wykrywania ataków DoS/DDoS i wykorzystującego sieci Bayesa.

## 2. Protokoły TCP/IP

Głównym elementem Internetu jest zestaw protokołów sieciowych zwanych jako TCP/IP (Transmission Control Protocol / Internet Protocol). Obecnie pod określeniem TCP/IP kryje się cały zestaw protokołów służących do transferu danych (IP, TCP, UDP), kontroli poprawności połączeń (ICMP), zarządzania siecią (SNMP), zdalnego włączania się do sieci (Telnet), usług aplikacyjnych służących do przesyłania plików (FTP) oraz innych. Rodzina tych protokołów jest używana zarówno w sieciach lokalnych jak i rozległych. Protokół TCP/IP opiera się na modelu odniesienia OSI [1], jednak jego architektura różni się od modelu odniesienia. Definiuje ona tylko cztery warstwy. Czworowarstwowy hierarchiczny model protokołów TCP/IP nazywany jest stosem protokołów.



Rysunek 1 Stos protokołów TCP/IP a model OSI

- Warstwa łącza danych zapewnia niezawodną transmisję danych od jednego węzła do drugiego, izolując wyższe warstwy od fizycznego nośnika informacji. Ta warstwa jest odpowiedzialna za formowanie ramek z pakietów i za bezbłędną transmisję ramek danych (Ethernet, Token Ring, Arcnet)
- Warstwa sieciowa dostarcza środków do ustanawiania, utrzymania i rozłączania połączeń sieciowych między systemami otwartymi, w których rezydują komunikujące się aplikacje, i odpowiada, za obsługę błędów komunikacji. Ponadto warstwa sieciowa jest odpowiedzialna za funkcje routingu, który wyznacza optymalną pod względem liczby połączeń drogę przesyłania pakietu przez sieć. Podstawowym protokołem tej warstwy jest IP
- Warstwa transportowa zapewnia przezroczysty transfer danych typu point-to-point. Dbą o kolejność pakietów otrzymywanych przez odbiorcę. Sprawdza poprawność (CRC)

przesyłanych pakietów i w przypadku ich uszkodzenia lub zaginięcia, zapewnia ich retransmisję

- Warstwa aplikacji - Warstwa ta świadczy usługi końcowe dla aplikacji, min.: udostępnianie zasobów (plików, drukarek). Na tym poziomie rezydują procesy sieciowe dostępne bezpośrednio dla użytkownika

## 2.1. Protokół IP

Protokół IP (Internet Protocol) [2] jest odpowiedzialny za przesyłanie pakietów, zwanych tutaj datagramami, pomiędzy użytkownikami sieci. Jest protokołem bezpołączeniowym, co oznacza, że w trakcie transmisji nie sprawdza się poprawności datagramów przesyłanych przez sieć. Nie ma, zatem gwarancji ich dostarczenia, ponieważ mogą one zostać po drodze zagubione przekłamate lub uszkodzone. Jego podstawowymi funkcjami są:

- Określanie struktury datagramu
- Określanie schematu adresacji
- Kierowanie ruchem datagramów w sieci
- Defragmentowanie i scalanie datagramów

Obowiązującą powszechnie wersją protokołu IP jest wersja 4, obecnie wprowadza się rozwiniętą względem poprzedniej, wersję 6 [3].

Nagłówek protokołu IP, jest przesyłany z każdym datagramem. Moduły zajmujące się transmisją wykorzystują go do transportu danych pomiędzy stacjami źródłową i docelową. Wiadomości zawarte w nagłówku IP, są również wykorzystywane przy dzieleniu i łączeniu pakietów danych podczas transmisji.

IP jest protokołem zawodnym. Jedynym kryterium pozwalającym sprawdzić poprawność przesyłania jest suma kontrolna nagłówka zawarta w polu Header Checksum. Jeżeli w trakcie transmisji został odkryty błąd to pakiet jest niszczone przez stację, która wykryła niezgodność. W takim przypadku nie ma żadnych powtórek transmisji i kontroli przepływu danych.





dzielone na pakiety o długości 512 bajtów, 60 to długość maksymalnego nagłówka IP, a pozostałe 4 bajty to margines dla innych protokołów. Najczęściej spotykaną długością nagłówka IP jest 20 bajtów. Pakiet przed wysłaniem do kolejnego węzła, musi być w danym węźle odebrany w całości, co prowadzi do powstawania opóźnień. Dlatego też pakiety wrażliwe na opóźnienia są krótkie.

- Identification [16 bitów]: Wartość ta jest ustawiana przez nadawcę i ma pomagać w identyfikacji fragmentów przy scalaniu pakietów.
- Flags [3 bity]: Flagi kontrolne:
  - Bit 0: zarezerwowany, musi mieć wartość 0
  - Bit 1: (DF) 0 = Można rozdrobnić pakiet, 1 = Nie można rozdrobnić
  - Bit 2: (MF) 0 = Ostatni fragment, 1 = Nie ostatni fragment
- Fragment Offset [13 bitów]: To pole określa, gdzie w oryginalnym pakiecie powinien być umieszczony dany fragment, powstały w wyniku podziału pakietu na części. Jednostką tutaj jest 8 bajtów (64 bity). Pierwszy fragment ma wartość offsetu zero.
- Time to Live [8 bitów]: To pole określa maksymalny czas przebywania pakietu w sieci. W momencie, gdy wartość zawarta w polu osiągnie zero, pakiet danych musi zostać zwrócony. Wartość pola jest modyfikowana w czasie przesyłania komunikatu w sieci. Jednostką są tu sekundy, ale każda z przetwarzających pakiet stacji ma obowiązek zmniejszyć wartość, o co najmniej jeden (nawet, gdy przetwarzanie zajęło mniej czasu). Działanie to ma spowodować wyeliminowanie pakietów niemożliwych do dostarczenia.
- Protocol [8 bitów]: To pole określa, który protokół został użyty na wyższym poziomie w przetwarzaniu danych pakietu.
- Header Checksum [16 bitów]: Suma kontrolna nagłówka danych IP. Ponieważ nagłówek może się zmieniać, suma jest obliczana w każdym węźle w sieci, do którego dotarł pakiet.
- Source Address [32 bity]: Adres nadawcy pakietu.
- Destination Address [32 bity]: Adres odbiorcy pakietu.
- Options: Dodatkowe informacje o pakiecie danych. Jest to opcjonalna część nagłówka protokołu IP i najczęściej nie jest dołączana do pakietu. Obecnie zdefiniowane opcje to:
  - Ograniczenia dotyczące bezpieczeństwa (dla aplikacji wojskowych)
  - Zapis trasy przebytej przez pakiet
  - Zapis jednocześnie drogi i czasu
  - Luźne wyznaczanie drogi przez nadawcę
  - Ścisłe wyznaczanie drogi przez nadawcę

## 2.2. Protokół ICMP

Protokół ICMP (Internet Control Message Protocol) [5] jest bezpołączeniowym protokołem kontrolnym. Przenosi on komunikaty kontrolne o stanie sieci i ma umożliwić routerom przekazywanie informacji o błędach i innych nagłych sytuacjach. Przykładem komunikatu jest np. Echo Request/Reply. Inne komunikaty zostaną umówione dokładniej przy okazji wykorzystywania ich do ataków DoS/DDoS. Format nagłówka protokołu jest następujący:

0								1								2								3							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7

Type	Code	Checksum
Unused		

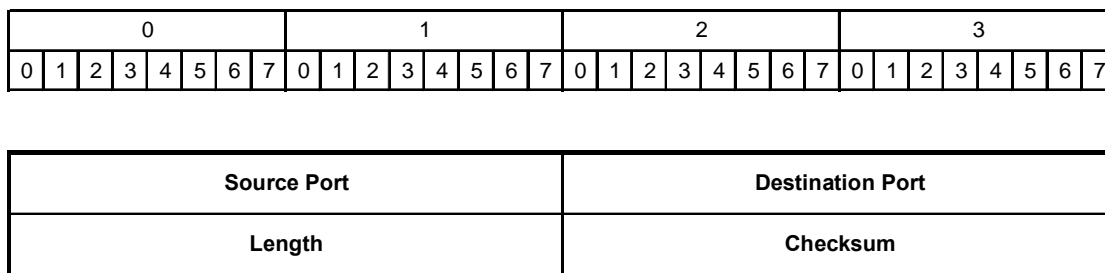
Rysunek 3 Nagłówek protokołu ICMP

- Type [8 bitów]: Określa typ komunikatu
- Code [8 bitów]: Specyfikuje dokładnie kod wiadomości w ramach danego typu
- Checksum [16 bitów]: Suma kontrolna całego pakietu ICMP

## 2.3. Protokół UDP

UDP (User Datagram Protocol) [4] jest protokołem warstwy trzeciej stosu protokołów, używanym do transportu danych w trybie bezpołączeniowym. Jak zostało to już wcześniej opisane, tryb bezpołączeniowy nie gwarantuje dostarczenia danych do odbiorcy. W momencie, gdy pakiet nie dotrze do odbiorcy, lub z wyliczonej sumy kontrolnej wyniknie, że pakiet jest uszkodzony, to UDP nie podejmie żadnych działań zmierzających do korekty lub retransmisji danych. Protokół UDP opracowano, by umożliwić aplikacjom bezpośrednie korzystanie z usług protokołu IP, dzięki dołączeniu do datagramów IP adresów portów komunikujących się aplikacji.

Nagłówek UDP zawiera między innymi numer portu źródłowego i numer portu docelowego. Obydwa numery portów są identyfikatorami programów ze znajdującej się wyżej warstwy aplikacji. W Internecie niektóre numery portów są zarezerwowane i wstępnie przypisane do tzw. dobrze znanych usług. I tak na przykład DNS ma przypisany port 53, a NTP [43] numer 123.



Rysunek 4 Nagłówek protokołu UDP

- Source Port [16 bitów]: Pole adresu portu źródłowego.
- Destination Port [16 bitów]: Pole adresu portu docelowego.
- Length [16 bitów]: Pole określające długość sumaryczną nagłówka i danych.
- Checksum [16 bitów]: Pole zawierające sumę kontrolną pakietu.

## 2.4. Protokół TCP

Protokół TCP (Transmission Control Protocol) [6] tworzy wirtualne połączenie pomiędzy dwoma maszynami umożliwiając na bezpieczny (bezbłędny) transfer danych. Odmienne niż UDP, TCP zapewnia wiarygodne połączenie dla wyższych warstw komunikacyjnych przy pomocy sum kontrolnych i numerowanie pakietów w celu weryfikacji wysyłki i odbioru. Brakujące pakiety są obsługiwane przez żądania retransmisji. TCP porządkuje też pakiety, które na przykład na skutek podróżowania różnymi ścieżkami, przybywają w nieprawidłowej kolejności.

Do wyspecyfikowania odbiorcy za pomocą 32-bitowej adresacji, wspieranej przez protokół IP, TCP dodaje mechanizm wyróżniający w każdym urządzeniu 65536 tzw. portów. Port jest logiczną jednostką, do której kierowane są informacje. Do każdego portu może być przypisany program odbierający wyłącznie informację skierowaną do danego portu. Istnieje grupa wyróżnionych numerów portów, dla których podano standardowy program obsługujący i rodzaj przesyłanych danych, np. przesyłanie plików FTP [7] – 20 i 21, Telnet [41] 23, HTTP [42] 80.

0								1								2								3							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7

Source Port										Destination Port									
Sequence Number																			
Acknowledgment Number																			
Data Offset		Reserved			URG	ACK	PSH	RST	SYN	FIN	Window								
Checksum										Urgent Pointer									
Options												Padding							

Rysunek 5 Nagłówek protokołu TCP

- Source Port [16 bitów]: Pole adresu portu źródłowego
- Destination Port [16 bitów]: Pole adresu portu docelowego
- Sequence Number [32 bity]: Pole zawierające numer sekwencji
- Acknowledgment Number [32 bity]: Pole zawierające numer odpowiedzi
- Data Offset [4 bity]: Pole wskazujące początek danych w pakiecie TCP i tym samym koniec nagłówka
- Reserved [6 bitów]: Zarezerwowane i nieużywane pole
- Flags [6 bitów]: Jednobitowe flagi, określające stan połączenia i umożliwiające jego zmianę
- Window [16]: Pole zawierające tzw. okno, czyli przedział numerów ACK, jaki nadawca jest w stanie zaakceptować.
- Checksum [16 bitów]: Suma kontrolna pakietu TCP
- Urgent Pointer [16 bitów]: Wskaźnik na dane natychmiastowe
- Options: Dodatkowe opcje pakietu
- Padding: Dopełnienie nagłówka do wielokrotności 32-bitowego słowa

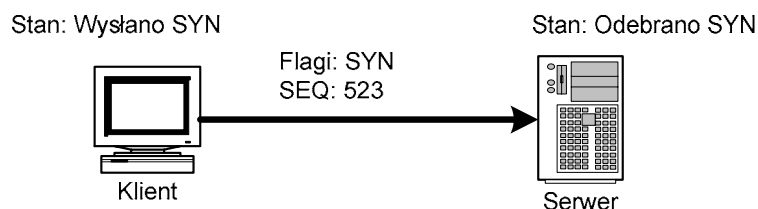
### 2.4.1. Nawiązywanie połączenia

Nawiązywanie połączenia TCP odbywa się w trzech fazach:



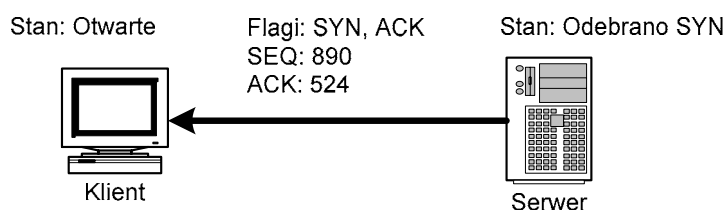
Rysunek 6 Nawiązywanie połączenia TCP – stan początkowy

1. Klient wysyła do serwera pakiet TCP z ustawionym bitem SYN oraz początkowym numerem sekwencji SEQ.



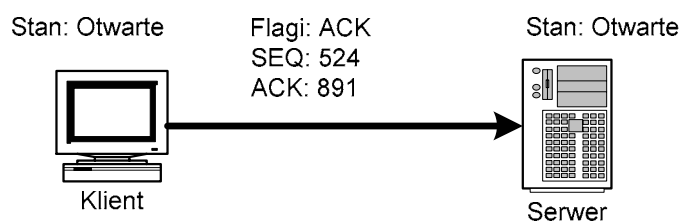
Rysunek 7 Nawiązywanie połączenia TCP – pierwsza faza

2. Serwer odsyła pakiet TCP z ustawionymi flagami SYN, ACK oraz numer sekwencji, od której będzie odliczał wysyłając dane.



Rysunek 8 Nawiązywanie połączenia TCP – druga faza

3. Klient wysyła pakiet z ustawioną flagą ACK.



Rysunek 9 Nawiązywanie połączenia TCP – trzecia faza

4. Połączenie jest otwarte i od tej chwili można wysyłać dane.

### 3. Ataki typu DoS/DDoS

Ataki typu DoS (Denial of Service czyli odmowy usługi) [44] mają na celu uniemożliwienie uprawnionym użytkownikom skorzystania z zasobów lub usług systemu komputerowego. Istnieje bardzo wiele odmian tego ataku - od przecięcia kabla zasilającego po skoordynowane uderzenie pakietami sieciowymi z tysięcy komputerów jednocześnie.

Urządzenia komputerowe do sprawnego działania potrzebują kilku podstawowych zasobów, takich jak czas procesora, powierzchnia dyskowa, pamięć RAM, przepustowość łącza. Ataki DoS starają się doprowadzić do zużycia bądź zablokowania tych zasobów, wykorzystując przy tym wiele z codziennie wykonywanych przez użytkowników czynności. Przepelniona skrzynka pocztowa i utracone na skutek tego listy to prosty przykład ataku DoS o nazwie Mial Bombing [8].

Można również utworzyć bardzo wiele małych plików, co np. w systemie Linux może doprowadzić do wyczerpania maksymalnej liczby węzłów (INODE) przechowujących informacje o plikach i w konsekwencji do niemożności stworzenia jakiegokolwiek następnego pliku. Kolejną formą ataku DoS jest zajęcie całego czasu procesora przez procesy jednego użytkownika (fork bomb), co uniemożliwia korzystanie z zasobów przez inne osoby.

Osobna grupa ataków DoS polega na zniszczeniu lub modyfikacji danych o konfiguracji programów i urządzeń. Intruz może zdalnie lub lokalnie zniszczyć pliki konfiguracyjne, blokując dostęp uprawnionym użytkownikom do danego programu (usługi) lub uniemożliwiając jego prawidłowe działanie. Na przykład modyfikując tablicę routingu może on zablokować całą sieć, a zmieniając wartość rejestru systemowego Windows - doprowadzić do wyłączenia niektórych usług lub do bardzo dziwnego, uniemożliwiającego pracę, zachowania się systemu. Istotnym środkiem obrony przed atakami DoS jest fizyczne zabezpieczenie sprzętu - zniszczenie kluczowego routera zablokuje skutecznie całą sieć na długi czas. [58]

Ataki sieciowe DoS nie wymagają lokalnego bezpośredniego dostępu do atakowanej maszyny, wykorzystują one luki w oprogramowaniu sieciowym oraz niedoskonałości w protokołach komunikacyjnych. Aplikacje sieciowe po otrzymaniu niepoprawnych, często specjalnie spreparowanych danych, przerywają działanie lub zajmują cały czas procesora, a nawet restartują maszynę. Bardzo często ataki DoS wykorzystują błędy w implementacji protokołów w systemach operacyjnych, np. tworzenie niewłaściwej tablicy ARP, nie sprawdzanie długości pakietów przed ich przetworzeniem (ataki Land, Teardrop, Ping of Death) czy nieprawidłowa obsługa składania pofragmentowanych pakietów IP.

DDoS, czyli Distributed Denial of Service jest udoskonaloną wersją ataku typu DoS - Denial of Service. Znacznie zmodyfikowaniu uległy głównie skuteczność oraz "bezpieczeństwo" agresora. O ile w metodzie DoS atak odbywa się z komputera agresora, o tyle atak DDoS

przeprowadzany jest w sposób rozproszony tzn. z wielu komputerów jednocześnie. Komputery te znajdują się w różnych lokalizacjach, a ich użytkownicy nie są świadomi tego, iż właśnie biorą udział w ataku na serwer internetowy. Aby komputer taki mógł wziąć udział w ataku musi być wcześniej zarażony odpowiednim programem złośliwym. Przeważnie są to konie trojańskie lub bomby logiczne, które dopiero na wyraźny sygnał od agresora uaktywniają się i rozpoczynają proces destrukcji. Wykrycie takiego programu złośliwego jest stosunkowo trudne ze względu na to, iż aktywuje się on tylko i wyłącznie w momencie ataku, po czym znów przechodzi w stan uśpienia. Programy tego typu mogą być bardzo inteligentne - po przeprowadzonym ataku starannie zacierają ślady swojej obecności w systemie nieświadomego użytkownika samoczynnie się deinstalując i kasując.

### 3.1. Fazy ataku

W zależności od rodzaju ataku, jego przebieg może być za każdym razem inny. Można jednak wyróżnić kilka faz ataku, które są wspólne dla większości z nich.

- **Faza I – Poszukiwania błędów**

Do przeprowadzenia skomasowanego i skutecznego ataku typu DDoS agresor potrzebuje armii maszyn, systemów operacyjnych, nad którymi będzie mógł przejąć kontrolę i wykorzystać do własnych celów. Dla dużych firm, portali i innych popularnych systemów internetowych nieprzerwana widoczność w sieci to ich być albo nie być. Ataki DoS/DDoS są dla nich dużym zagrożeniem, dlatego inwestują środki w dobrą infrastrukturę sieciową:

- Równoległe łącza do wielu dostawców często o przepustowościach setek Mbit/s.
- Wydajne routery, które są zaprojektowane do obsługi o wiele większego ruchu niż aktualnie występujący. Routery te często odpowiadają za balansowanie ruchu na wszystkich dostępnych łączach. Zaatakowanie jego z nich powoduje przeniesienie użytkowników na inne.
- Wydajne zapory ogniowe renomowanych firm.
- Systemy IDS (Intrusion Detection Systems), czyli systemami wykrywania włamań. [20]
- Właściwa konfiguracja wszystkich elementów sieci oraz aktualne oprogramowanie.
- Całodobowy monitoring, ze strony własnego personelu technicznego, jak i administratorów po stronie dostawców ISP.

Atak na takie instytucje wymaga sieci DDoS o liczbie setek tysięcy maszyn. W początkach ataków typu DDoS, sieć była budowana z maszyn pracujących pod kontrolą systemów klasy Unix. Jednak ich stosunkowo mała liczba oraz dbałość administratorów (w większości



przypadków) o ich bezpieczeństwo powodowało, że zbudowanie wystarczająco dużej sieci nie było zadaniem prostym. Wtedy agresorzy zwrócili się w kierunku systemów biurkowych firmy Microsoft (Windows 95, Windows 98, Windows Me, Windows 2000, Windows XP, Windows 2003). W stosunku do maszyn pracujących pod systemem Unix, mają one więcej zalet w punkcie widzenia agresora:

- Popularność. Sytemu klasy Windows są zainstalowane na ponad 95% [9] komputerów PC i liczba podłączonych do sieci maszyn działających pod tym systemem jest ogromna.
- Niewiedza użytkowników. Przeciętny użytkownik tego systemu posiada nikłą wiedzę na temat bezpieczeństwa sieciowego. Dodatkowo użytkownicy, często z lenistwa nie aktualizują swoich systemów, nie instalują prywatnych zapór ogniowych innych elementów zwiększających bezpieczeństwo.
- Duża liczba krytycznych błędów – Statystyki błędów pokazują, że w każdym kwartale jest wykrywanych średnio po kilka błędów, które mają status krytyczny i umożliwiają przejęcie zdalnej kontroli.
- Słabe bezpieczeństwo – Domyślnie w tej klasie systemów zwykły użytkownik ma prawa administratora i może zrobić wszystko ze swoim komputerem. Uruchamiane przez niego programy również nie podlegają ograniczeniom. Takie środowisko jest wprost wymarzone dla wirusów i robaków.

W tej pierwszej fazie, agresor albo intensywnie poszukuje błędów w systemach operacyjnych, albo czeka spokojnie aż inni je znajdą.

## • Faza II – Kodowanie

Kiedy odpowiedni błąd zostanie znaleziony, agresor musi stworzyć odpowiedni program (tzw. exploit), aby go wykorzystać. Do jego kodu, następnie dodawane są następne moduły, z których najczęściej możemy spotkać:

- Moduł do skanowania sieci w poszukiwaniu następnych ofiar
- Moduł ukrywający narzędzie w systemie
- Moduły do infekcji innymi drogami (poczta elektroniczna, dyskietki itp.)
- Moduł atakujący DDoS
- Moduł autodestrukcji
- Moduł komunikacji z innymi agentami i agresorem

Mniej pracowici wandalę, często składają robaka z gotowych klocków napisanych przez innych.

- **Faza III – Budowanie sieci**

Kiedy narzędzie jest gotowe, wystarczy że agresor zarazi kilka komputerów na początek, a te z kolei zainfekują następne. W zależności od sposobu infekowania, poszukiwania innych maszyn oraz oczywiście popularności błędu, liczba maszyn w sieci DDoS może w ciągu kilku godzin osiągnąć nawet kilka milionów.

- **Faza IV – Atak**

Atak na wybrany cel odbywa się albo o ściśle określonym czasie, już wybranym podczas kodowania, albo agenci komunikują się między sobą i bezpośrednio, lub pośrednio otrzymują komendę od agresora.

## 3.2. Historia ataków DoS/DDoS

Tabela 1 Historia ataków DoS/DDoS [10]

1995	Atakujący na pogawędkach sieciowych powodują zawieszenie komputerów wysyłając tzw. Ping of Death.
1998	Specjaliści do spraw bezpieczeństwa wykrywają coraz więcej prób ataków typu ICMP SMURF na serwery stron WWW. Ataki powodują blokowanie tych serwerów.
1998	Tysiące komputerów w NASA, Marynarce Wojennej USA oraz na uniwersytetach zostaje odciętych przez zalewające ich dane. Źródło ataku nie zostało wykryte.
Luty 2000	Atak DDoS powoduje wyłączenie Yahoo.com, eBay, Amazon.com i wiele innych serwisów na kilka godzin. FBI aresztuje 15 letniego chłopca o pseudonimie „Mafiaboy”, który wykorzystał laboratorium Uniwersytetu Kalifornijskiego do przeprowadzenia ataku.
Maj 2001	Robak Code Red miał na celu zaatakowanie witryny Białego Domu. Atak prawie 250 tys. maszyn nie udał się ze względu na przeniesienie serwisu na inny adres IP.
Październik 2002	Atak DDoS przeciwko 13 głównym serwerom DNS, które przechowują główne domeny. Pomimo dużej skali ataku oraz czasu jego trwania nie

	doszło do przerywania komunikacji. Według ekspertów dłuższy atak, mógł zachwiać działaniem światowej sieci.
Marzec 2003	Robak o nazwie Deloader zainfekował prawie 18 tys. komputerów, zamieniając je w aktywną sieć DDoS.
Sierpień 2003	Robak Blaster i kilka jego wariantów zainfekowało około 500 tys. komputerów. Celem robaka był atak na stronę firmy Microsoft oferującą uaktualnienia. Atak nie powiódł się po tym jak Microsoft wyłączył ją na czas ataku.
Styczeń 2004	Wirus o nazwie MyDoom bardzo szybko zainfekował około 1 miliona komputerów. W szczytowym okresie, szacuje się, że co czwarty wysłany e-mail w sieci pochodził od wirusa. Celem pierwszej wersji robaka był atak na stronę główną firmy SCO, natomiast druga wersja atakowała również stronę Microsoftu. Atak na stronę SCO powiódł się w 100% i firma ta musiała zmienić adres swojej głównej strony. Microsoft nie odczuł ataku wirusa.

## 4. Podział i klasyfikacja ataków typu DDoS/DoS

Jako główne czynniki decydujące o klasyfikacji ataków typu DDoS/DoS pod uwagę zostały wzięte środki służące do przygotowania i przeprowadzenia ataku, charakterystyka samego ataku, oraz efekt, jaki on wywiera na ofiarę. Liczne kryteria klasyfikacji zostały zaznaczone pogrubionym tekstem. [21]

### 4.1. Klasyfikacja na podstawie stopnia automatyzacji

Podczas przygotowań do ataku, agresor musi znaleźć odpowiednią ilość właściwych maszyn-agentów. Bazując na **stopniu automatyzacji** ataku, wyróżniamy: ręczne, półautomatyczne i automatyczne ataki DDoS/DoS.

#### 4.1.1. Ataki ręczne

Tylko pierwsze ataki typu DDoS należały do tej kategorii. Agresor skanował zdalne komputery w poszukiwaniu luk, włamywał się na nie, instalował odpowiednie narzędzia oraz ręcznie włączał atak. Wszystkie te akcje zostały później zautomatyzowane i rozwinięte do półautomatycznych działań, kategorii, do której należy większość dzisiejszych ataków.

#### 4.1.2. Półautomatyczne ataki

W półautomatycznych atakach sieć DDoS składa się z głównego węzła (master, handler) oraz agentów (slave, daemon). Agresor dostarcza zautomatyzowany skrypt, który:

- Skanuje komputery w poszukiwaniu potencjalnych ofiar
- Dokonuje włamań na nich
- Instaluje oprogramowanie do przeprowadzenia ataku

W dalszej kolejności, agresor korzystając z głównego węzła sieci wybiera typ ataku i adres ofiary a następnie wydaje komendę do tego rozpoczęcia.

Na podstawie **mechanizmu komunikacji** wykorzystywanego przez agentów i główny węzeł sterujący możemy podzielić półautomatyczne ataki na ataki: z bezpośrednią komunikacją oraz pośrednią komunikacją.

##### 4.1.2.1. Ataki z bezpośrednią komunikacją

Podczas ataków z bezpośrednią komunikacją, agent musi znaleźć główny węzeł sieci, aby mógł się z nim połączyć. Jest to przeważnie osiągane przez zapisanie adresu IP głównego węzła w kodzie agenta. Zwykle każdy agent komunikuje swoją gotowość, pozwalając na

zapisanie swojego numeru IP w celu późniejszej komunikacji. Oczywistą wadą tej metody jest to, że odkrycie jednej zainfekowanej maszyny ujawnia całą sieć DDoS, oraz że agenci nasłuchują na połączenia, przez co są narażeni na szybkie wykrycie przez skanery sieciowe.

#### 4.1.2.2. Ataki z pośrednią komunikacją

Zastosowanie pośredniej komunikacji ma na celu zmniejszenia prawdopodobieństwa wykrycia sieci DDoS. Jednym z przykładów tej metody jest komunikacja z użyciem serwerów IRC [45]. Kanały IRC mogą zastępować główny węzeł sieci dostarczając atakującemu wystarczającą anonimowość. Jako że agenci nawiązują połączenia do standardowej usługi, taka transmisja może być trudno wykrywalna wśród autoryzowanego ruchu. Agenci nie nasłuchują na żadnym porcie, przez co nie zostaną wykryci przez skanery sieciowe. Wykonywanie poleceń odbywa się poprzez odpowiednie, wcześniej zdefiniowane komendy pojawiające się na kanałach IRC. Odkrycie jednego z agentów w większości wypadków nie demaskuje reszty sieci DDoS oraz agresora. IRC jest jednym z przykładów i nic nie stoi na przeszkodzie użycia innych podobnych usług (np. pogawędek na portalach).

#### 4.1.3. Automatyczne ataki

Aby uniknąć komunikacji pomiędzy agentami i agresorem, część z nich stosuje zautomatyzowaną fazę atakującą. Czas ataku, rodzaj, długość oraz adres ofiary są zaprogramowane w kodzie agenta. Zakodowanie dokładnych danych sprawia, że sieć DDoS ma określony z góry cel. Jednakże, mechanizmy propagacji zostawiają zwykle tylną furtkę umożliwiając łatwą zmianę parametrów szturm.

Zarówno półautomatyczne, jak i automatyczne ataki wykorzystują inne komputery do stworzenia sieci DDoS. Bazując na **strategii skanowania** w poszukiwaniu nowych agentów możemy wyróżnić: skanowanie losowe, z listą, topologiczne, permutacyjne oraz skanowanie lokalnej sieci.

##### 4.1.3.1. Strategie skanowania

- Skanowanie losowe

Podczas skanowania losowego każdy zarażony komputer próbuje przypadkowe numery IP z własnej przestrzeni adresowej – ich generatory liczb losowych zostały zainicjowane różną wartością (ang. seed). Taki sposób potencjalnie wywoła duże zwiększenie ruchu, ponieważ wiele agentów może próbować zarażać na raz jedną maszynę. Code Red (CRv2) przeprowadzał losowe skanowanie [11].

- Skanowanie z listą

W tej metodzie skanowania agenci skanują w poszukiwaniu ofiar wykorzystując dostarczoną z zewnątrz listę potencjalnych ofiar. Kiedy nowa maszyna zostaje zainfekowana, agent wysyła jej połowę swojej listy, zatrzymując drugą połowę. Technika ta umożliwia bardzo dużą szybkość propagacji (wykładniczą) i brak kolizji podczas skanowania. Taka lista może zostać pobrana np. z stron typu netscan.org, czyli adresów sieci umożliwiających wysyłaniem komunikatów echo na adres rozgłoszeniowy (atak typu SMURF – patrz pkt. 6.3.1.1).

- Skanowanie topologiczne

Topologiczne skanowanie wykorzystuje informacje znalezione na zainfekowanym komputerze w celu określenia następnych celów ataku. Wszystkie robaki pocztowe stosują topologiczne skanowanie korzystając z książki adresowej ofiary.

- Skanowanie permutacyjne

Podczas skanowania permutacyjnego wszystkie zainfekowane maszyny współdzielą pseudo-losową permutację przestrzeni adresów. Każdy adres IP jest mapowany na odpowiedni indeks w tej permutacji. Nowy agent zaczyna skanowanie używając indeksu obliczonego z jego numeru IP jako punktu startowego. Kiedy jednak wykryje już zainfekowaną maszynę, wybiera nowy punkt startowy. Taki algorytm pozwala na pół-skoordynowane, pełne skanowanie wykorzystujące element losowości. Na razie metoda ta nie została zaimplementowana w żadnym z robaków. [12]

- Skanowanie sieci lokalnej

Skanowanie sieci lokalnej może stanowić dodatek do każdej metody wymienionej powyżej, aby sprawdzić podatność komputerów będących w tej samej podsieci (wyznaczonej przez numer IP i jego maskę sieciową). Używając tej techniki pojedynczy wątek skanujący może zarazić wiele maszyn znajdujących za zaporą ogniową. Code Red II [11] oraz Nimda Worm [13] skanowały sieci lokalne.

Ze względu na **mechanizm propagacji kodu** można rozróżnić: ataki z centralną propagacją kodu, łańcuchową oraz autonomiczną.

## 4.2. Propagacja kodu

- Ataki z centralną propagacją kodu

Wykorzystując centralną propagację, kod znajduje się na jednym bądź kilku wybranych serwerach. Po zarażeniu nowej ofiary, kod ten jest pobierany poprzez protokół HTTP, bądź FTP (TFTP). Metodę tę stosował robak Iloft [14].

- Ataki z łańcuchową propagacją kodu

W łańcuchowej propagacji, kod jest pobierany bezpośrednio z maszyny atakującej. Zainfekowana maszyna staje się automatycznie źródłem dla następnego kroku propagacji. Wybranie tego sposobu zwiększa szanse powodzenia zbudowania potężnej sieci DDoS poprzez wyeliminowanie jednego słabego punktu. Pewnym zagrożeniem może być ryzyko uszkodzenia kodu i jego dalszej niekontrolowanej propagacji. Ta metoda została zaimplementowana w robakach Morris [15] i Ramen [16].

- Autonomiczna propagacja

Autonomiczna propagacja unika pobierania kodu robaka poprzez jego bezpośrednie wstrzyknięcie podczas fazy zarażania. Code Red [11], Warhol Worm [12] i wiele robaków pocztowych używa autonomicznej propagacji.

### 4.3. Klasyfikacja na podstawie wykorzystywanych słabości ofiary

Rozproszony atak typu odmowa usługi wykorzystuje różne strategie, aby uniemożliwić uprawnionym użytkownikom dostęp do usług. Biorąc pod uwagę **słabości ofiary wykorzystywane podczas** szturmu możemy wyróżnić: ataki na protokoły oraz siłowe ataki.

#### 4.3.1. Ataki na protokół

Ataki na protokół wykorzystują specyficzną własność, bądź lukę niektórych protokołów zainstalowanych u ofiary powodując wyczerpanie jej zasobów. Jako przykład można podać atak typu SYN Flood (patrz pkt. 6.1), wielokrotne żądanie skryptu CGI lub atak na serwer autoryzacji.

Podczas pierwszego z nich w wyniku zalewu pakietami typu SYN następuje zapelnienie pamięci wykorzystywanej przy przechowywaniu części sesji półotwartych połączeń i niemożność nawiązania nowych. Wielokrotne żądanie skryptu CGI wykonującego złożone operacje arytmetyczne lub na bazie danych skutkuje zajętością procesorów maszyny na poziomie 100%. Podobnie działa atak na serwer autoryzacji, w którym jest on zalewany prośbą o uwierzytelnienie.

#### 4.3.2. Ataki siłowe

Ataki siłowe wykorzystują efekt saturacji łącza ofiary, czyli zajęcie całego pasma jakim jest ona połączoną ze swoim providerem. Jak wiadomo połączenie każdej sieci z Internetem jest ograniczone parametrami przepustowości – wychodzącej do internetu (TX) i przychodzącej z Internetu (RX). Pasma może wahać się od wartości 56 Kbit/s (połączenie modemowe), aż do kilkuset megabitów na sekundę (łącza ATM). Na obecną chwilę średniej wielkości firma w

Polsce dysponuje łączem o przepustowości 1Mbit. W takiej sytuacji wystarczy, że agresor wygeneruje do ofiary ruch o natężeniu 1,2Mbit/s i łącze ofiary zostanie zajęte w 100%. W takich warunkach po obu stronach łącza tworzą się dość duże kolejki pakietów czekających na transmisję. Część z nich dotrze z dużym opóźnieniem (dla użytkownika będzie wyraźne spowolnienie pracy), natomiast część może zostać odrzucona będą w kolejce. Przy jeszcze większym ruchu, rzędu (1,5Mbit/s) oczekiwanie na połączenia wydłuży się do takiego okresu, że dla użytkownika serwer praktycznie przestanie być osiągalny.

Ze względu na **rodzaj ruchu w stosunku do usług ofiary** możemy ataki siłowe podzielić na: możliwe do wyfiltrowania i nie możliwe do wyfiltrowania.

#### 4.3.2.1. Ataki możliwe do wyfiltrowania

Jeśli atak siłowy używa usług lub protokołów, które nie mają krytycznego znaczenia dla ofiary, taki ruch można zablokować na zaporze ogniowej. Przykładem takiego ataku jest zalew pakietów UDP albo na przykład zalew serwera WWW pakietami ICMP.

#### 4.3.2.2. Ataki nie możliwe do wyfiltrowania

W sytuacji, kiedy agresor generuje ruch do krytycznych usług ofiary, problemem staje się jego wyfiltrowanie. Nie jest możliwe zablokowanie portu, bądź protokołu na zaporze ogniowej, bo to wprowadzi odepnie atak, ale również użytkowników. Przykładem tego rodzaju ataku jest zalewanie serwera WWW żadaniami głównej strony czy atak na serwer DNS (prośba o rozwinięcie domeny).

Linia podziału pomiędzy atakami na protokół i atakami siłowymi jest dość cienka. Atak na protokół również mają na celu saturację łącza ofiary, a źle zaprojektowana własność protokołu jest często wykorzystywana do tworzenia powielaczy (ang. reflectors) i przeprowadzania ataków siłowych np. SMURF (patrz pkt. 6.3.1.1). Różnica polega w dużej mierze na tym, że w przypadku ataku na protokół można zminimalizować ryzyko poprzez odpowiednią konfigurację, natomiast w przypadku ataku siłowego niewiele można zrobić w celu uchronienia się przed nim.

### 4.4. Klasyfikacja ze względu na dynamikę ataku

Ze względu na **dynamikę ataku**, wyróżniamy: ataki ciągłe oraz ataki zmienne

#### 4.4.1. Ataki ciągłe

Większość znanych ataków typu DDoS/DoS mała charakter ciągły. Kiedy sieć agentów była już gotowa, następowała komenda do ataku i wszystkie zainfekowane komputery generowały



pakiety z maksymalną siłą. Taki nagły potop pakietów bardzo szybko paraliżował usługi ofiary, przez co był natychmiast wykrywany.

#### 4.4.2. Ataki zmienne

Ataki zmienne są bardziej wyrafinowane, przez co trudniej przebiega ich detekcja oraz reakcja. Ze względu na **rodzaj zmiany siły ataku** wyróżniamy podkategorie: wzrastającej siły oraz zmiennej siły.

##### 4.4.2.1. Ataki ze wzrastającą siłą

Ataki, które cechują się stopniowo wzrastającą siłą powoli wykradają zasoby ofiary. Wzrost może następować przez dłuższy czas usypiając czujność administratorów oraz powodując, że urządzenia zaprojektowane do wykrywania anomalii nie wykryją tego ataku.

##### 4.4.2.2. Ataki z zmienną siłą

W atakach, które są przeprowadzane ze zmienną siłą, natężenie generowanego ruchu może naprzemiennie wzrastać i maleć utrudniając wykrycie ataku. Na samym końcu tej podkategorii znajdują się atak pulsacyjny. Powódź pakietów jest co jakiś czas ponawiana i stopowana. Jeśli te działania są równoległe w przypadku wszystkich agentów, serwis ofiary jest okresowo blokowany. Jeśli jednakże agenci są podzieleni na grupy, które są między sobą tak skoordynowane, że zawsze jedna z nich jest aktywna, wtedy serwis ofiary jest zablokowany cały czas.

#### 4.5. Klasyfikacja ze względu na skutek ataku DoS/DDoS

W zależności od **skutku**, jaki wywołuje atak możemy wyróżnić: ataki powodujące całkowitą blokadę ofiary lub ataki obniżające poziom usług ofiary.

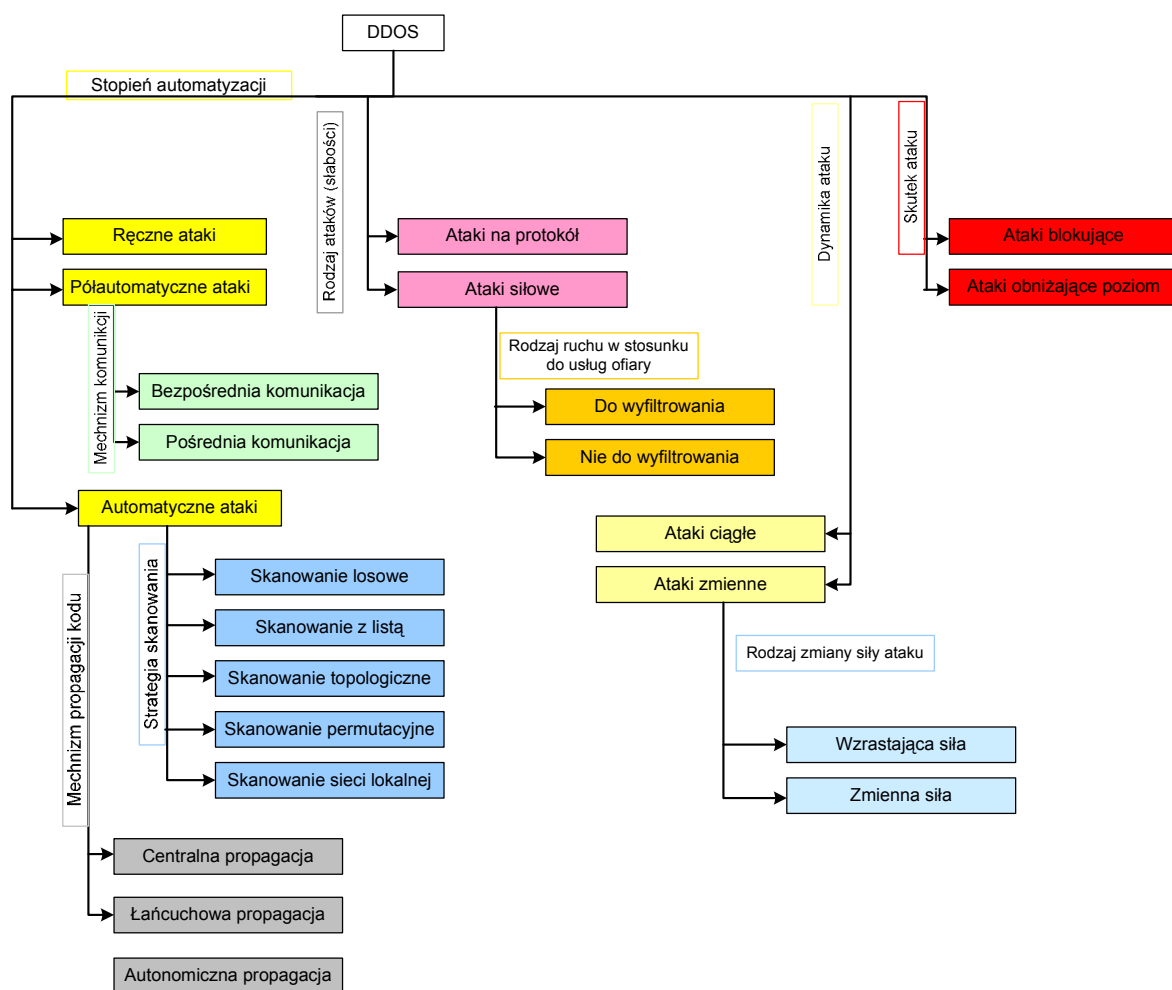
##### 4.5.1. Ataki blokujące

Celem tego ataku jest całkowite odcięcie klientów od danego serwisu. Prawie wszystkie znane do tej pory ataki DDoS/DoS miały taki cel.

##### 4.5.2. Ataki obniżające poziom

O wiele bardziej finezyjne są ataki, które nie powodują całkowitej blokady usług ofiary. Celem ich natomiast jest pochłonięcie pewniej części (przeważnie stałej) zasobów. Przez to są trudne do wykrycia i mogą przez długi okres czasu pozostać niezauważone. Z drugiej strony koszty dla ofiary mogą być znaczne. Dla przykładu pochłonięcie 30% zasobów ofiary spowoduje, że w okresach dużego obciążenia część klientów nie będzie mogła skorzystać z

usługi i poziom jej się obniży. To może następnie prowadzić do odejścia części klientów niezadowolonych z usług. Jest to już konkretna strata. Innym kosztem takiego utajnionego ataku mogą być niepotrzebne wydatki na zwiększenie zasobów, przepustowości itp.



Rysunek 10 Podział i klasyfikacja ataków typu DoS/DDoS [21]

## 5. Podział i klasyfikacja metod obrony

Poważny kłopot, jaki sprawiają ataki typu DoS/DDoS spowodował opracowanie wielu strategii obrony. Część z nich adresowana jest do konkretnych ataków, jak ataki na serwer WWW czy usługę autoryzacji. Inne natomiast próbują rozwiązać ogólny problem typowych ataków DoS. Większość zaproponowanych rozwiązań wymaga pewnych własności środowiska, aby osiągnąć swoją najwyższą wydajność, podczas gdy w pozostałych warunkach ich skuteczność jest przeciętna.

Aby przejść dalej, trzeba zrozumieć jedną rzecz: **nie ma dzisiaj złotego środka przeciwko atakom typu DDoS**. Aby móc częściowo obronić się przed atakami należałoby wdrożyć wiele ze strategii opisanych poniżej równocześnie. Różne kryteria klasyfikacji zostały zaznaczone pogrubionym tekstem. [21]

Biorąc pod uwagę **poziom aktywności** strategii obrony przed atakami typu DDoS możemy wyróżnić: prewencyjne oraz reaktywne mechanizmy.

### 5.1. Mechanizmy prewencyjne

Celem mechanizmów prewencyjnych jest po pierwsze wyeliminowanie możliwości przeprowadzenia ataku DDoS oraz po drugie przygotowanie potencjalnych ofiar na przetrwanie ataku, bez odcięcia potencjalnych klientów naszych usług. Z uwagi na właśnie na dwa **cele**, możemy strategie obrony podzielić na *zapobiegające atakom* oraz *zapobiegające odmowie usługi*.

#### 5.1.1. Zapobieganie atakom

Mechanizmy zapobiegające atakom mają na celu taką zmianę konfiguracji systemu oraz sieci, aby wyeliminować możliwość ataku. Ze względu na **rodzaj** tych zmian wyróżniamy: zmiany w bezpieczeństwie systemów oraz zmiany w bezpieczeństwie protokołów.

##### 5.1.1.1. Zmiany w bezpieczeństwie systemów

Zmiany w bezpieczeństwie systemów mają na celu zwiększenie ich odporności przed nieuprawnionym dostępem, usunięcie błędów i luk, aby nie można było ich użyć w niewłaściwym celu. Siła ataków DDoS leży w ogromniej liczbie maszyn, który generują jednocześnie ruch. Jeśli teoretycznie wszystkie one zostały by wcześniej zabezpieczone, ewentualny agresor straci swoją moc. Z drugiej strony dziurawe systemy często stają się w pierwszej kolejności ofiarami ataków DoS, kiedy atakujący uzyska do nich pełny dostęp i zniszczy dane. Do mechanizmów zwiększających bezpieczeństwo systemów możemy zaliczyć:

- Monitorowanie dostępu do maszyny [17]
- Systemy automatycznie aktualizujące oprogramowanie
- Zapory ogniowe [18]
- Programy antywirusowe [19]
- Systemy wykrywania włamań IDS [20]
- Listy krytycznych zasobów

Historia bezpieczeństwa komputerowego wskazuje, że spełnienie wszystkich tych zaleceń nie ochroni nas w 100% przed atakiem, ale może w dużym stopniu zmniejszyć jego prawdopodobieństwo.

#### 5.1.1.2. Zmiany w bezpieczeństwie protokołów

Adresatem tej strategii są źle zaprojektowane protokoły. Wiele protokołów zawiera operacje, które są łatwe i tanie dla klientów oraz bardzo kosztowne dla serwera. Takie protokoły mogą być z łatwością wykorzystane, aby wyczerpać zasoby serwera przez rozpoczęcie wielu jednoczesnych transakcji. Klasycznymi przykładami są ataki typu SYN Flood (6.1), atak na serwer autoryzacji oraz atak z wykorzystaniem fragmentacji, gdzie serwer otrzymuje setki źle skonstruowanych pakietów zmuszające go do marnowania swoich zasobów na próbach ich złożenia. Do zmian w bezpieczeństwie protokołów możemy zaliczyć:

- Bezpieczne projektowanie protokołów (zasoby serwera są udostępnione tylko uwierzytelnionym klientom) [22]
- Przerzucenie części kosztu transakcji na klienta
- Wdrożenie odpowiednich serwisów proxy (np. składających połączenia TCP) [23]
- Wyłączenie niektórych opcji protokołów np. fragmentacji IP

Wdrożenie odpowiednich polityk bezpieczeństwa systemów oraz protokołów diametralnie zmniejsza szanse stania się ofiarą ataku typu DDoS/DoS. Działania te powinny stać się podstawą i być połączone z innymi strategiami obrony.

#### 5.1.2. Zapobieganie odmowie usługi

Zapobieganie odmowie usługi jest strategią, w której staramy się, aby nasze serwery przetrwały atak DDoS/DoS, przez co nasi klienci nie zostali odcięci od usług. Realizuje się to poprzez nałożenie ścisłych polityk wykorzystania zasobów albo poprzez duży ich zapas, który może być awaryjnie wykorzystany podczas ataków. Z uwagi na **metodę prewencyjną** możemy wyróżnić: strategie limitowania zasobów oraz mnożenia zasobów.

### 5.1.2.1. Limitowanie zasobów

W metodzie limitowania zasobów każdy użytkownik otrzymuje ich część na podstawie swoich praw oraz zachowania. Taki mechanizm gwarantuje sprawiedliwy dostęp dla normalnie pracujących użytkowników. Mechanizm ten jest często łączony z systemami automatycznego uwierzytelniania, aby uniemożliwić kradzieże tożsamości.

### 5.1.2.2. Mnożenie zasobów

Mnożenie zasobów opiera się dużej ich nadmiarowości w stosunku do normalnych potrzeb. W przypadku ataku, te zasoby mogą być wykorzystane i uniemożliwić atak typu odmowa usługi. Najprostszym przykładem jest farma serwerów, połączonych bardzo szybkimi łączami, na których ruch jest wyrównywany, aby zapewnić optymalne wykorzystanie. Taka strategia obrony oczywiście podnosi poprzeczkę - liczby maszyn, jaka musi uczestniczyć w ataku, aby był on skuteczny. Ta metoda, mimo, że nie jest idealna może być dość skuteczną obroną dla firm, które są w stanie ponieść jej koszt. Przez dłuższy czas firma Microsoft w miarę skutecznie broniła się korzystając z tej metody.

## 5.2. Mechanizmy reaktywne

Mechanizmy reaktywne koncentrują się na obronie naszych serwerów, kiedy atak już trwa. Aby móc się skutecznie ochronić atak należy *wyryć* oraz na niego *zareagować*.

### 5.2.1. Wykrywanie ataków

Celem mechanizmów wykrywania ataków jest identyfikacja ich tak szybko jak to tylko możliwe oraz niski stopień fałszywych alarmów. Kiedy już uda się wykryć atak, można poznać charakterystykę powodujących go pakietów a następnie zastosować odpowiednie strategie odpowiedzi.

Klasyfikujemy **strategie wykrywania ataków** na detekcje: na podstawie wzorca, wykrywania anomalii, metodę hybrydową oraz na podstawie zewnętrznych sygnałów.

#### 5.2.1.1. Mechanizmy wykrywania ataków na podstawie wzorca

Mechanizm wykrywający ataki na podstawie wzorca korzysta z bazy sygnatur znanych ataków. Każda transmisja jest monitorowana i porównywana do symptomów ataków zarejestrowanych wcześniej. Baza jest okresowo aktualizowana o nowe sygnatury ataków. Oczywiście wadą tej metody jest to, że wykrywana ona tylko znane ataki i jest bezsilna wobec nowych rodzajów, a nawet odmian starych ataków niepasujących do przechowywanych wzorców. Z drugiej strony znane ataki są prosto i pewnie wykrywane a liczba fałszywych alarmów jest niska.

#### 5.2.1.2. Wykrywanie ataków na podstawie anomalii

Mechanizm wykrywania ataków na podstawie anomalii posiada model normalnego zachowania systemu, czyli np. przeciętny poziom ruchu, czy średnie obciążenie systemu. Aktualny stan systemu jest okresowo porównywany do modelu, aby wykryć odstępstwa od normy. Zaletą tego rodzaju mechanizmu jest możliwość wykrywania nieznanych ataków, jednak posiada on kilka kłopotliwych własności:

- Wartość progu. Alarm jest podnoszony, kiedy aktualny stan systemu, bądź sieci odbiega od modelu o zdefiniowaną wartość progową. Jeśli wartość progowa będzie na niską spowoduje to wiele fałszywych alarmów. Z kolei, jeśli ustawimy za dużą wartość tym samym zredukujemy czułość mechanizmu.
- Aktualizacja modelu. Obciążenie systemów oraz sieci zmienia się z upływem czasu i model musi uwzględniać te zmiany. Zwykle realizuje się to poprzez automatyczną aktualizację korzystając ze zgromadzonych statystyk. Problem mogą być różne próby ataków, ataki pulsacyjne, które zniekształcą te statystyki. Również zastosowanie ataku, którego siła rośnie z czasem bardzo powoli, spowoduje, że ten atak może zostać wykryty z dużym opóźnieniem lub nie wykryty w ogóle.

#### 5.2.1.3. Hybrydowe wykrywanie ataków

Systemy implementujące hybrydowe wykrywanie ataków korzystają z bazy sygnatur znanych ataków oraz potrafią wykrywać nowe ataki przez analizę anomalii. Nowe ataki są następnie ładowane do bazy. Metoda ta jest dość dobra i z powodzeniem jest wykorzystywana w wielu systemach IDS [20].

Jeśli te systemy są w pełni zautomatyzowane, poprawne wyekstraktowanie sygnatury ataku może być naprawdę trudne. System powinien być ostrożny, aby nie dać się oszukać przez agresora i zakwalifikować normalnego ruchu jako sygnaturę ataku, gdyż wtedy sam stanie się narzędziem DoS.

#### 5.2.1.4. Wykrywanie ataków na podstawie zewnętrznych sygnałów

Mechanizm ten nie obserwuje bezpośrednio symptomów ataku, lecz opiera się na zewnętrznych agentach przekazujących informację o ataku i jego charakterystykę. Przykładem takiego mechanizmu jest śledzenie wsteczne z użyciem ICMP [24] (Traceback). Mechanizm ten zakłada dodatkową funkcjonalność, która pozwoliłaby na odkrycie źródła (DDoS – źródło) ataku, nawet wtedy, kiedy adres nadawcy był by sfałszowany. Idea jest dość prosta i polega na automatycznym wysyłaniu przez routery do odbiorcy pakietów ICMP z adresem IP poprzedniego routera (od który ten otrzymał pakiet). Takie pakiety byłyby wysyłane, co ok 20 000 pakietów, przez co nie miałyby wpływu na obciążenie sieci. Podczas ataku DDoS/DoS ofiara otrzymywałaby jednocześnie pakiety od wszystkich routerów

znajdujących się pomiędzy nią a atakującym. Koncepcja ta wymagałaby jednak skoordynowanych zmian na większości routerów w sieci Internet.

### 5.2.2. Reakcja na atak

Celem reakcji na atak jest oczywiście zmniejszenie jego negatywnych skutków dla serwera oraz sieci, oraz korzystających z niego użytkowników. Na podstawie **strategii reakcji** wyróżniamy: identyfikację agentów agresora, limitowanie ruchu, filtrowanie oraz rekonfiguracja.

#### 5.2.2.1. Identyfikacja agentów agresora

Identyfikacja agentów agresora pozwala na ustalenie maszyn, które są odpowiedzialne za atak typu DoS/DDoS. Czynność ta jest przeważnie wykonywana równolegle z innymi reakcjami. Techniki identyfikacji agentów to na przykład wspomniana w poprzednim rozdziale technika śledzenia wstecznego ICMP.

#### 5.2.2.2. Limitowanie ruchu

Limitowanie ruchu ma na celu ograniczenie napływu powodzi pakietów do ofiary. Limitowanie ruchu jest zazwyczaj stosowane, kiedy nie ma pewności, co do wykrytego ataku czy jest on faktycznie atakiem a nie normalnym działaniem. Wadą tego działania jest przedostawanie się części pakietów do atakowanego serwera, przez co ataki na bardzo dużą skalę mogą być nadal skuteczne.

#### 5.2.2.3. Filtrowanie ruchu

Mechanizm filtrowania ruchu używa charakterystyki dostarczonej przez moduł wykrywający, aby odfiltrować kompletnie strumień pakietów. Na rynku jest wiele produktów, które implementują ten mechanizm – dynamicznej zapory ogniowej [28]. Dopóki mechanizm wykrywania nie jest dostatecznie pewny istnieje ryzyko zablokowania ruchu od uprawnionych użytkowników. W najgorszym przypadku, sprytny agresor może użyć tego mechanizmu jako DoS.

#### 5.2.2.4. Rekonfiguracja

Ostatnim z proponowanych działań jest rekonfiguracja sieci ofiary. Zmiany mogą zawierać: zmianę topologii sieci, dołożenie dodatkowych zasobów lub izolację atakowanych maszyn.

Reakcja obronna na atak typu DoS/DDoS może być wykonana przez samą ofiarę lub w kooperacji z innymi podmiotami. Bazując na **stopniu kooperacji** rozróżniamy pomiędzy: autonomicznymi, kooperatywnymi oraz współzależnymi mechanizmami.

### 5.2.3. Mechanizmy autonomiczne

Działanie mechanizmów autonomicznych ogranicza się tylko do jednego punktu w sieci i jest zazwyczaj przeprowadzane przez samą ofiarę ataku. Zapory ogniowe [29], systemu IDS to proste przykłady takich narzędzi.

### 5.2.4. Mechanizmy kooperatywne

Autonomiczne wykrycie ataku i reakcja może być o wiele skuteczniejsza, kiedy będzie przeprowadzona przez wiele podmiotów jednocześnie. Przykładem technologii, w której wymagana jest kooperacja jest mechanizm o nazwie Pushback [25]. Wykrywa on ataki DoS/DDoS poprzez obserwacje natężenia ruchu na routerach. Jego charakterystyka jest następnie wykorzystywana do konstruowania reguł limitujących jego przepływ. Następnie routery wymieniają informację pomiędzy sobą i wszystkie z nich aplikują te reguły. W tym momencie atak jest prawie całkowicie blokowany.

### 5.2.5. Mechanizmy współzależne

Mechanizmy współzależne nie mogą działać autonomicznie – potrzebują innych podmiotów do wykrycia ataku i reakcji na niego. Wspomniany wcześniej mechanizm wstecznego śledzenia ICMP jest dobrym przykładem takiego narzędzia. Ma on jedynie sens w sytuacji, kiedy wszystkie routery po drodze będą go obsługiwać.

Biorąc pod uwagę **miejsce reakcji** na atak rozróżniamy pomiędzy mechanizmami: u ofiary, w sieci pośredniej oraz sieci agresora.

#### 5.2.5.1. Mechanizmy obrony w sieci ofiary

Czynności wykonywane w sieci ofiary mają na celu ochronę jej zasobów, serwerów, urządzeń sieciowych przez atakami typu DoS/DDoS. Historycznie, większość mechanizmów obronnych jest ulokowana właśnie u ofiary, gdyż to ona cierpi najbardziej i jest zmotywowana, aby poświęcić swoje środki na ochronę. Limitowanie zasobów, wzmocnienie ochrony przed dziurawymi protokołami są przykładem tych mechanizmów.

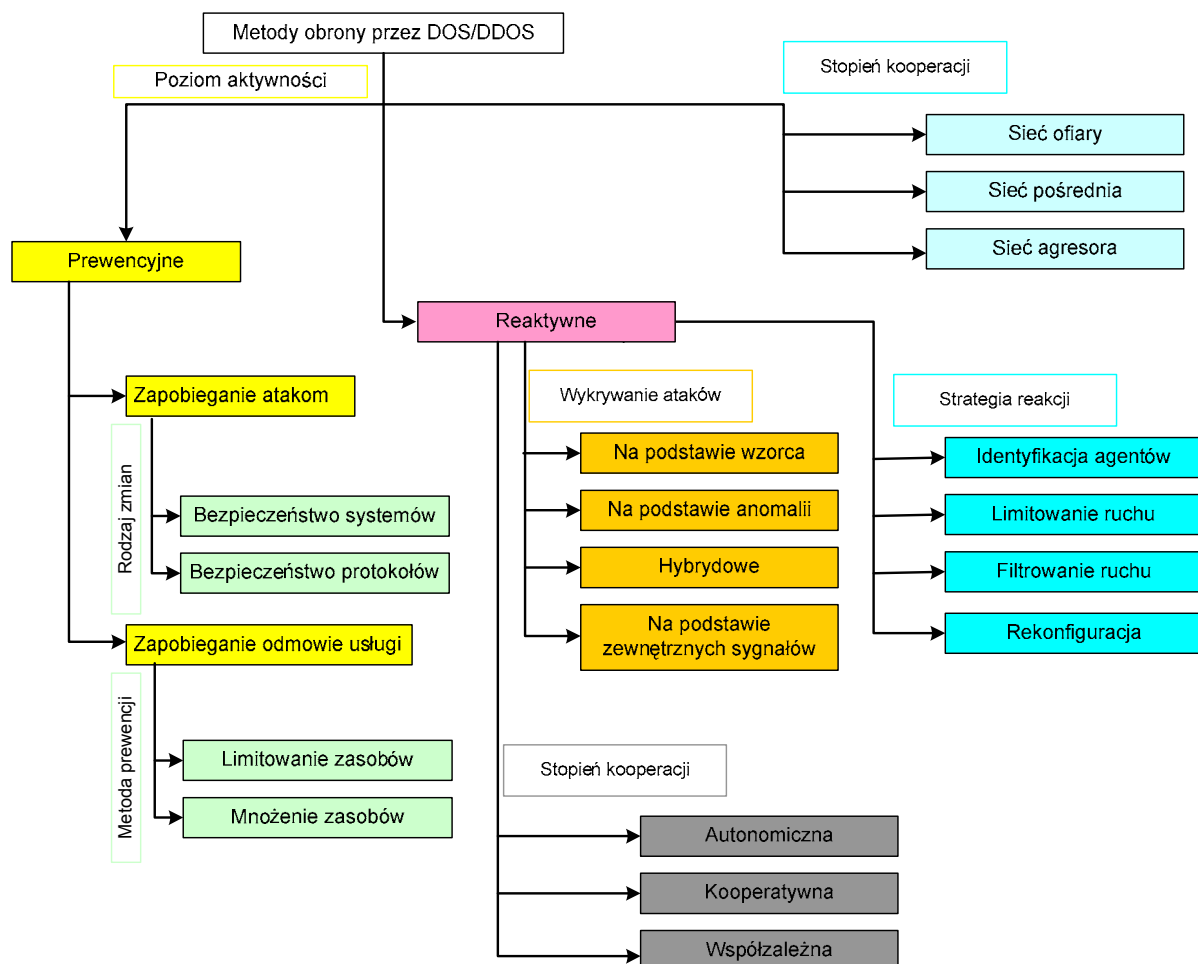
#### 5.2.5.2. Mechanizmy obrony w sieci pośredniej

Mechanizmy obrony przed ataki DoS/DDoS są wdrażane w całej infrastrukturze pośredniej pomiędzy ofiarą a potencjalnym agresorem. W założeniach ofiara podczas ataku może automatycznie skontaktować się np. z routerem i poprosić o pomoc w identyfikacji ruchu. Pushback i śledzenie wsteczne są przykładami takich mechanizmów.

#### 5.2.5.3. Mechanizmy obrony w sieci agresora



Celem mechanizmów wdrażanych w sieci agresora jest uniemożliwienie lub utrudnienie mu przeprowadzenia ataku typu DoS/DDoS. Mechanizmy te są skuteczne i pożądane, jednak ich stopień wdrożenia jest dość niski z uwagi na niechęć do ponoszenia dodatkowych kosztów. Przykładem takiego mechanizmu jest filtrowanie ruchu wychodzącego z danej sieci z adresem źródłowym nieprzydzielonym do niej, co jest jednak typowym działaniem zapór ogniowych – kontrola niespójności kierunkowych.



Rysunek 11 Podział i klasyfikacja metod obrony przed atakami typu DoS/DDoS [21]

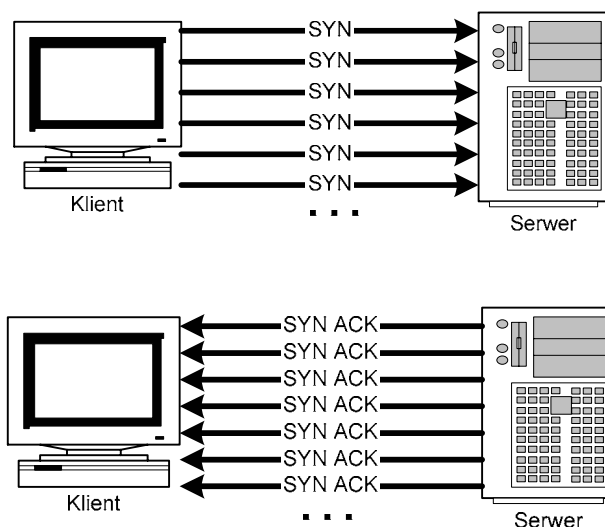
## 6. Popularne ataki typu DoS i DDoS

### 6.1. Powódź pakietów SYN - SYN Flood

Jednym z pierwszych ataków DoS była powódź pakietów typu SYN (ang. SYN Flood). SYN (będące skrótem od ang. Synchronize) jest nawą flagi w pakiecie TCP – jej zapalenie w pierwszym docierającym pakiecie jest konieczne w celu nawiązania połączenia (2.4.1). Powódź tych pakietów powoduje znaczną alokację zasobów serwera i może uniemożliwić pracę uprawnionym użytkownikom. Ten typ ataku może być wykorzystywany w celu blokowania wszystkich usług działających na warstwie TCP, jednak najczęściej był wykorzystywany w celu blokowania serwerów WWW (protokół HTTP).

#### 6.1.1. Anatomia ataku

Atak typu SYN Flood wykorzystuje podstawową słabość protokołu TCP/IP - otwarcie połączenia w trzech krokach. Normalna sekwencja nawiązania połączenia (przedstawiona w 2.4.1) zostaje przerwana w drugim kroku tj. klient nie odsyła serwerowi pakietu ACK. Po stronie serwera połączenie ma stan połowicznie otwartego – serwer oczekuje na pakiet ACK.



Rysunek 12 Przebieg ataku typu SYN Flood

Dla takich połączeń systemy operacyjne serwerów mają przeznaczone określoną ilość zasobów. Czas oczekiwania na pakiet ACK jest wcześniej zdefiniowany (zwykle kilkadziesiąt sekund). Jeśli jest on wystarczająco długi, a napływ pakietów SYN duży, nastąpi wyczerpanie zasobów i serwer nie będzie mógł obsłużyć uprawnionych klientów.

Przeprowadzenie ataku typu SYN Flood, jest dość łatwe z dwóch powodów:

- Wielkość pakietu SYN minimalnie może wynosić 40 bajtów (40 bajtów nagłówek + 0 bajtów danych), co umożliwia atakowanie korzystając z dość słabych łącz internetowych
- Atakujący nie potrzebuje odpowiedzi serwera (pakietu z flagami SYN, ACK), dlatego w większości przypadków w adres nadawcy pakietów SYN jest wstawiana przypadkowa wartość. Jeśli wylosowany adres IP jest przypisany do działającego urządzenia sieciowego, na wysłany pakiet SYN-ACK urządzenie to powinno odpowiedzieć pakietem z flagą RST – który trafia do atakowanego dodatkowo powodując saturację jego łącza.

Zakładając, że atakujący mógłby wykorzystać 100% przepustowości swojego łącza i wielkość pakietu 40 bajtów, można policzyć intensywność ataku:

Tabela 2 Przepustowość łącza i liczbę pakietów SYN

Modem 56 Kbit	99 pakietów SYN/s
ISDN 128 Kbit	400 pakietów SYN/s
Frame Relay 1Mbit/s	3200 pakietów SYN/s

Jak każdy inny atak, również i SYN Flood może mieć charakter rozproszony.

### 6.1.2. Przykłady ataków

Głośne przypadki ataków typu SYN Flood miały miejsce w lutym 2000 roku. Jedne z największych firm internetowych CNN, Yahoo, eBay i Amazon stały się celem ataku przez okres kilka dni. Po kilku miesiącach śledztwa aresztowany został 15 letni Kanadyjczyk o pseudonimie „Mafiaboy”. Poszkodowani szacowali swoje straty na wiele milionów dolarów. [26] [27]

### 6.1.3. Obrona

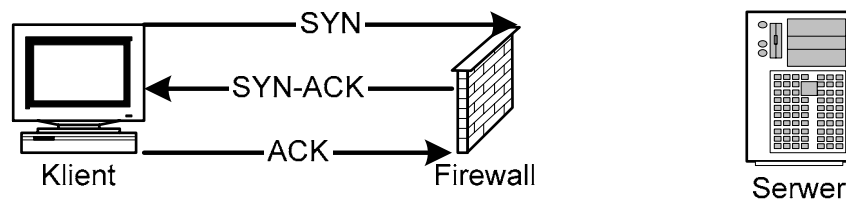
#### 6.1.3.1. Ogólne metody obrony

Po dość spektakularnych atakach na największe w owym czasie firmy internetowe na rynku powstało wielkie zapotrzebowanie na urządzenia sieciowe bądź mechanizmy, które pozwolą przetrwać atak typu SYN Flood. Część z tych metod zostało zaimplementowanych i weszło do użycia. Oto kilka z nich:

- Filtr pakietów SYN (SYN threshold): pozwala na ustanowienie pewnej ilości niekompletnych połączeń, a następnie odrzuca pakiety SYN, jeśli liczba niekompletnych połączeń osiągnie limit. Limity są przeważnie wyznaczane na źródłowy adres IP. Jest to

najprostsza metoda, która posiada jednak dużo wad. Jedną z nich jest to, że w przypadku losowanego generowania adresu źródłowego jest ona nieskuteczna. Jednakże metoda ta została zaimplementowana w kilku produktach.

- SYNDefender [23]: jest to technika zaimplementowana w firewallach firmy Check Point Software Technologies, w produkcie Firewall-1. Jej działanie polega na buforowaniu 3 fazowego nawiązywania połączenia. Firewall przekazuje pakiety, tylko wtedy, kiedy połączenie zostanie nawiązane.



Rysunek 13 Sposób działania mechanizmu SYNDefender

- SYN cookies [31]: celem tej metody jest wyeliminowanie konieczności przechowywania informacji o niekompletnych połączeniach poprzez zawarcie potrzebnych informacji („ciasteczku”) w pakiecie odsyłanym do klienta (SYN-ACK). Kiedy klient odpowie pakietem ACK, ciasteczko powraca i serwer jest w stanie wydobyć informacje potrzebne od odbudowania połączenia. Metoda ta została zaimplementowana w kilku systemach klasy UNIX (m.in. w Linux’ie).

### 6.1.3.2. Linux

System Linux bardzo często obecnie instalowany na wszelakiego typu routerach, mostkach i innych urządzeniach, już od wersji kernela 2.0.x posiada wbudowany mechanizm ochrony przed atakiem typu SYN Flood. Nazywa się on SYN cookies [31]. Aby go uaktywnić należy, należy się upewnić, że opcja:

```
Networking options ---> IP: TCP syncookie suport
```

jest włączona. Następnie można normalnie skompilować i zainstalować kernel (więcej informacji o kompilowaniu i instalacji serwera można uzyskać w Kernel HOWTO [30]). Większość dostępnych dystrybucji na rynku (RedHat, Debian, Mandrake, SuSe) posiada domyślnie tę opcję kompilowaną. Nawet po wkompilowaniu tego mechanizmu jest on domyślnie wyłączony. Spowodowane jest to tym, że przy bardzo obciążonych systemach mogą się pojawić problemy z jej działaniem, dlatego uruchomienie jej musi być świadomą decyzją administratora systemu, który będzie mógł monitorować jej działanie. Aby włączyć ten mechanizm należy wydać polecenie:

```
echo „1” > /proc/sys/net/ipv4/tcp_syncookies
```

Mechanizm ochrony przed atakami typu SYN Flood został wymyślony i zaimplementowany w latach 1996/1997, chociaż sama idea „ciasteczek” w tej dziedzinie jest nieco starsza. Z punktu widzenia atakowanego systemu, najlepiej było by, gdyby odpowiedź SYN-ACK na pakiet SYN nie była pamiętana. Wtedy pamięć i inne zasoby byłyby alokowane tylko po dostaniu ostatniego pakietu 3 stopniowej sekwencji z flagą ACK. Jednakże w przychodzącym pakiecie SYN jest kilka informacji, które system (serwer) powinien zapamiętać. (np. MSS - Minimum Segment Size). Dlatego w tej metodzie obrony numer sekwencji, obliczany jest przez serwer w ściśle określony sposób – kodując w ten sposób potrzebne informacje. Kiedy klient odsyła trzeci pakiet, wartości pół nagłówka TCP są odpowiednio dekodowane (można powiedzieć, że wcześniejsze pakiety są „odtworzane”). Różnica pomiędzy numerem sekwencyjnym klienta i serwera jest następująca:

- Najstarszych 5 bitów:  $t \bmod 32$ , gdzie  $t$  jest licznikiem czasowym zwiększanym co 64 s
- Następne 3 bity: zakodowana wartość MSS wybrana przez serwer w odpowiedzi na MSS klienta
- Młodsze 24 bity: zakodowane przez serwer adres IP klienta, IP Serwera, port źródłowy klienta, port docelowy serwera oraz operatorów

#### 6.1.3.3. Windows 2000/XP/2003

Ten najpopularniejszy system również ma wbudowane podstawowe mechanizmy obrony przed atakami typu DoS (SYN Flood). Zgodnie z opisem z MSDN [32] wszystkie te wartości ustawia się w następującym kluczu rejestru:

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services

- Nazwa wartości: SynAttackProtect  
Klucz: Tcpip\Parameters  
Typ wartości: REG\_DWORD  
Zakres prawidłowych wartości: 0,1  
Domyślna wartość: 0

Poniżej znajdują się parametry, których można używać z tą wartością rejestru:

- 0 (wartość domyślna): Ustawienie parametru SynAttackProtect na 0 zapewnia zwykłą ochronę przed atakami typu SYN.
- 1: Ustawienie parametru SynAttackProtect na 1 zapewnia lepszą ochronę przed atakami typu SYN. Parametr ten powoduje, że protokół TCP dopasowuje retransmisję pakietów SYN-ACK. Jeśli parametr SynAttackProtect ma wartość 1, to w przypadku rozpoznania ataku SYN obowiązują krótsze limity czasu odpowiedzi na zapytania połączeń. W

celu rozpoznania ataku system Windows wykorzystuje następujące wartości:

- TcpMaxPortsExhausted
  - TCPMaxHalfOpen
  - TCPMaxHalfOpenRetried
- Nazwa wartości: EnableDeadGWDetect  
Klucz: Tcpip\Parameters  
Typ wartości: REG\_DWORD  
Zakres prawidłowych wartości: 0, 1 (fałsz, prawda)  
Domyślna wartość: 1 (prawda)

Poniżej zestawiono parametry, których można używać z tą wartością rejestru:

- 1: Ustawienie parametru EnableDeadGWDetect na 1 powoduje, że protokół TCP zezwala na wykrywanie bram nieaktywnych. Gdy wykrywanie bram nieaktywnych jest włączone, a w przypadku wielu połączeń występują trudności, protokół TCP może zwrócić się do protokołu internetowego (IP) o zmianę bramy zapasowej. Bramy zapasowe można zdefiniować w sekcji Zaawansowane okna dialogowego konfiguracji protokołu TCP/IP w aplecie Sieć w Panelu sterowania.
- 0: Zaleca się ustawienie parametru EnableDeadGWDetect na 0. Jeśli parametr ten nie będzie miał wartości 0, atak może zmusić serwer do przełączenia bram, a w szczególności do uaktywnienia bramy niepożądaney.

- Nazwa wartości: EnablePMTUDiscovery  
Klucz: Tcpip\Parameters  
Typ wartości: REG\_DWORD  
Zakres prawidłowych wartości: 0, 1 (fałsz, prawda)  
Domyślna wartość: 1 (prawda)

Poniżej zestawiono parametry, których można używać z tą wartością rejestru:

- 1: Ustawienie parametru EnablePMTUDiscovery na 1 powoduje, że protokół TCP próbuje wykryć na ścieżce do hosta zdalnego maksymalną jednostkę transmisji (MTU) albo największy rozmiar pakietu. Wykrywając jednostkę MTU na ścieżce i ograniczając do jej wielkości rozmiary własnych segmentów, protokół TCP może wyeliminować fragmentację na routerach wzdłuż ścieżki łączącej sieci o różnych jednostkach MTU. Fragmentacja ma negatywny wpływ na przepływność protokołu TCP.

- 0: Zaleca się ustawienie parametru `EnablePMTUDiscovery` na 0. W takim wypadku dla wszystkich połączeń, które nie są nawiązywane przez hosty w lokalnej podsiaci jest używana jednostka MTU o wielkości 576 bajtów. Jeśli zostanie wybrana inna wartość niż 0, atakujący będzie mógł posłużyć się bardzo małą wartością jednostki MTU i przepełnić stos.

- Nazwa wartości: `KeepAliveTime`  
 Klucz: `Tcpip\Parameters`  
 Typ wartości: `REG_DWORD` - czas w milisekundach  
 Zakres prawidłowych wartości: 1 - 0xFFFFFFFF  
 Domyślna wartość: 7 200 000 (dwie godziny)

Parametr ten określa, jak często protokół TCP usiłuje sprawdzić, czy bezczynne połączenie jest wciąż aktywne, wysyłając pakiet utrzymania aktywności. Jeśli komputer zdalny jest wciąż osiągalny, potwierdza pakiet utrzymania aktywności. Pakiety utrzymania aktywności nie są wysyłane domyślnie. Ustawieniem zalecanym jest 300 000 (5 minut).

- Nazwa wartości: `NoNameReleaseOnDemand`  
 Klucz: `Netbt\Parameters`  
 Typ wartości: `REG_DWORD`  
 Zakres prawidłowych wartości: 0, 1 (fałsz, prawda)  
 Domyślna wartość: 0 (fałsz)

Parametr ten wprowadzono po to, aby umożliwić administratorowi ochronę komputera przed złośliwymi atakami polegającymi na wysyłaniu żądań zwolnienia nazw. Zaleca się ustawienie parametru `NoNameReleaseOnDemand` na 1.

Tabela 3 Rekomendowane wartości rejestru dla systemu Windows [32]

Nazwa klucza	Rekomendowana wartość
<code>SynAttackProtect</code>	2
<code>TcpMaxPortsExhausted</code>	1
<code>TcpMaxHalfOpen</code>	500
<code>TcpMaxHalfOpenRetried</code>	400
<code>TcpMaxConnectResponseRetransmissions</code>	2
<code>TcpMaxDataRetransmissions</code>	2
<code>EnablePMTUDiscovery</code>	0

KeepAliveTime	300000 (5 minut)
NoNameReleaseOnDemand	1

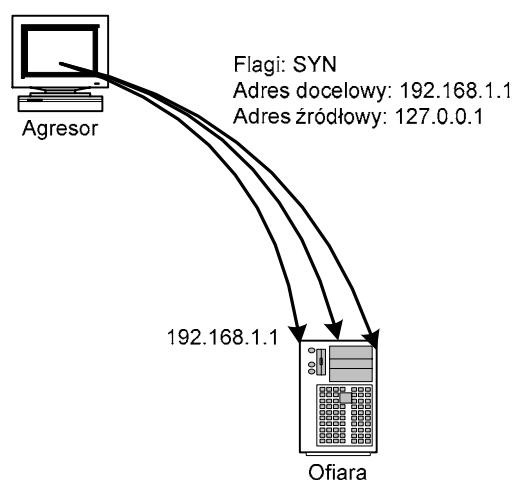
## 6.2. Atak typu LAND

### 6.2.1. Anatomia ataku

Atak typu land [33], nazywany jest również loopback SYN, jest odmianą ataku typu SYN Flood. W tej metodzie cała transmisja odbywa się po tzw. interfejsie loopback (lokalny adres komputera o numerze IP 127.0.0.1), co uniemożliwia oglądanie wysyłanych pakietów narzędziami sieciowymi. Atak ten wykorzystuje IP spoofing [46] - komputer włamywacza udaje uprawniony komputer, używając jego adresu IP.

Sam atak polega na przekierowaniu nawiązywanego połączenia TCP. Włamywacz musi jedynie odpowiednio rozpocząć inicjację połączenia TCP, czyli wysłać pakiet do atakowanego komputera z ustawionym bitem SYN i z jego własnym adresem jako adresem źródłowym. Atakowany komputer po otrzymaniu pakietu z flagą SYN wysyła w odpowiedzi (sam do siebie, po interfejsie loopback) pakiet z ustawionymi bitami SYN i ACK. Nie przerywa on połączenia, ponieważ otrzymuje pakiet z flagą ACK. Problem polega na tym, że nie jest sprawdzane ustawienie bitu SYN - nie byłoby kłopotu, gdyby pakiet z flagą ACK i SYN został odrzucony, a połączenie zamknięte. Ze względu na niezgodność numerów sekwencji otrzymanego pakietu, serwer próbuje od nowa nawiązać połączenie przez wysłanie kolejnego pakietu z ustawionymi bitami SYN oraz ACK. Prowadzi to do nieskończonej pętli, w której serwer utrzymuje dialog sam ze sobą.

Przebieg ataku LAND jest następujący:



Rysunek 14 Przebieg ataku typu LAND

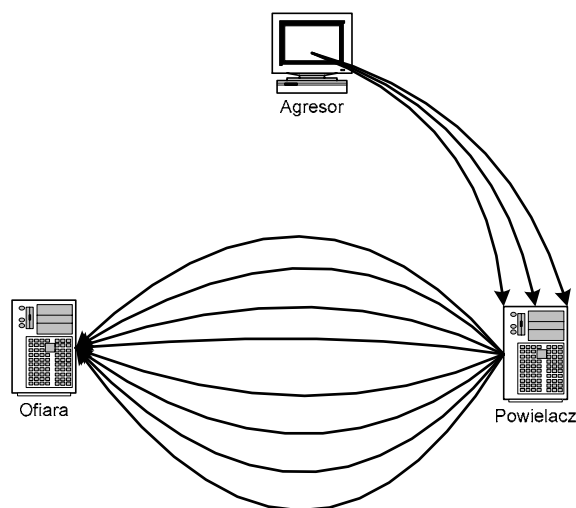


## 6.2.2. Obrona

Atak typu LAND pojawił się w roku 1997 i był poważnym problemem. Wiele systemów operacyjnych było na niego podatnych, jego zastosowanie przeważnie powodowało zawieszenie się urządzenia. Wśród nich były m.in. BSDI 2.1, FreeBSD, HP-UX, MacOS, NEXTSTEP, OpenBSD, Solaris, SunOS, Windows 95. Na dzień dzisiejszy takiego zarażenia już nie ma, atak jest w większości przypadków blokowany na firewallach, również same systemy operacyjne nie są na niego podatne (poprawiono implementacje stosu TCP/IP).

## 6.3. Ataki z wykorzystaniem powielaczy

Z punktu widzenia osoby chcącej przeprowadzić atak typu DoS wygenerowanie wystarczającego ruchu stanowi problem. Osoby te przeważnie mają dostęp do łącz szerokopasmowych jednak o niskiej przepustowości w stosunku do ofiary. Mając w domu łącze DSL o przepustowości 512 Kbit/s – od Internetu i tylko 128 Kbit/s do Internetu trudno atakować portal, który dysponuje sumaryczną przepustowością na poziomie kilkudziesięciu lub kilkuset Mbit/s. Zdobycie praw administratora na serwerach firm posiadających odpowiednią infrastrukturę (łącza) nie jest w większości wypadków ani sprawą prostą ani łatwą. Dużo prościej jest wykorzystać podstawowe usługi protokołów TCP/IP, które poprzez złą konfigurację mogą być wykorzystywane do atakowania innych serwerów. Przykład wykorzystania powielaczy można zobaczyć na poniższej ilustracji



Rysunek 15 Sposób wykorzystania przykładowego powielacza

- Klient wysyła pakiety ze sfałszowanym adresem ofiary do powielacza
- Powielacz odpowiada dużą ilością pakietów, bądź dużą ich wielkością do ofiary

Sens korzystania z powielacza ma jedynie sens wtedy, kiedy ruch generowany przez niego będzie kilka-kilkakrotnie większy od wielkości ruchu agresor-powielacz.

### 6.3.1. Wykorzystanie protokołu ICMP

Protokół komunikatów kontrolnych internetu ICMP [5] (ang. Internet Control Message Protocol) powstał, aby umożliwić routerom oznajmianie o błędach oraz udostępnianie informacji o niespodziewanych sytuacjach. Chociaż protokół ICMP powstał, aby umożliwić routerom wysyłanie komunikatów, to jednak każda maszyna może wysyłać komunikaty ICMP do dowolnej innej. Protokół ICMP jest traktowany jako wymagana część warstwy IP i musi być realizowany przez każdą implementację IP.

Są dwie różne drogi wykorzystania ICMP jako powielacza: użycie komunikatów zaprojektowanych jako żądanie/odpowiedź (np. ICMP echo), albo wysyłanie ruchu, który wygeneruje pakiet ICMP (w odpowiedzi na powstałe pewne problemy podczas przesyłania ruchu). W pierwszej kategorii możemy znaleźć komunikaty:

- Echo (służy do testowania kanałów komunikacyjnych pomiędzy dwoma komputerami – komunikaty wysyła polecenie ping)
- Timestamp (wykorzystywany do synchronizacji zegarów komputerów w sieci)
- Address mask (komunikaty te używane są do otrzymania maski podsieci, w której znajduje się komputer. ICMP Address Mask Request może być wysłany bezpośrednio do urządzenia, które udzieli tej informacji lub też jest rozprzestrzeniany w całej sieci lokalnej)
- Information Request (za ich pomocą komputer może uzyskać adres IP w sieci, w której się znajduje)

Natomiast w drugiej grupie można wyróżnić następujące komunikaty:

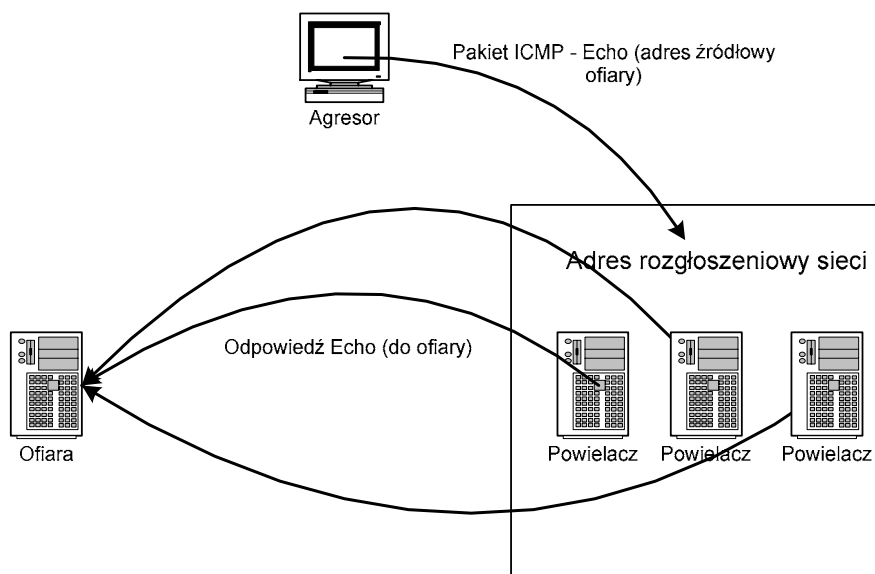
- Unreachable (komunikat oznacza, że pakiet z jakiś powodów nie mógł dotrzeć do adresata)
- Source quench (komunikat generowany w sytuacji, gdy bufor odebranych danych wypełni się i pakiet, który się w nim nie zmieści, zostanie porzucony)
- Redirect (gdy tylko router zidentyfikuje nadawcę używającego niewłaściwej trasy dla swoich pakietów, po przetworzeniu pakietu wysyła do nadawcy datagram informujący o istnieniu lepszej trasy dla jego pakietów)
- Time exceeded (komunikat ten jest wysyłany przez router albo bramę w sytuacji gdy napotkają one pakiet z parametrem TTL równym 0)
- Parameter problem (problem z przetworzeniem wysłanego parametru)

### 6.3.1.1. Atak typu SMURF

#### 6.3.1.1.1. Anatomia ataku

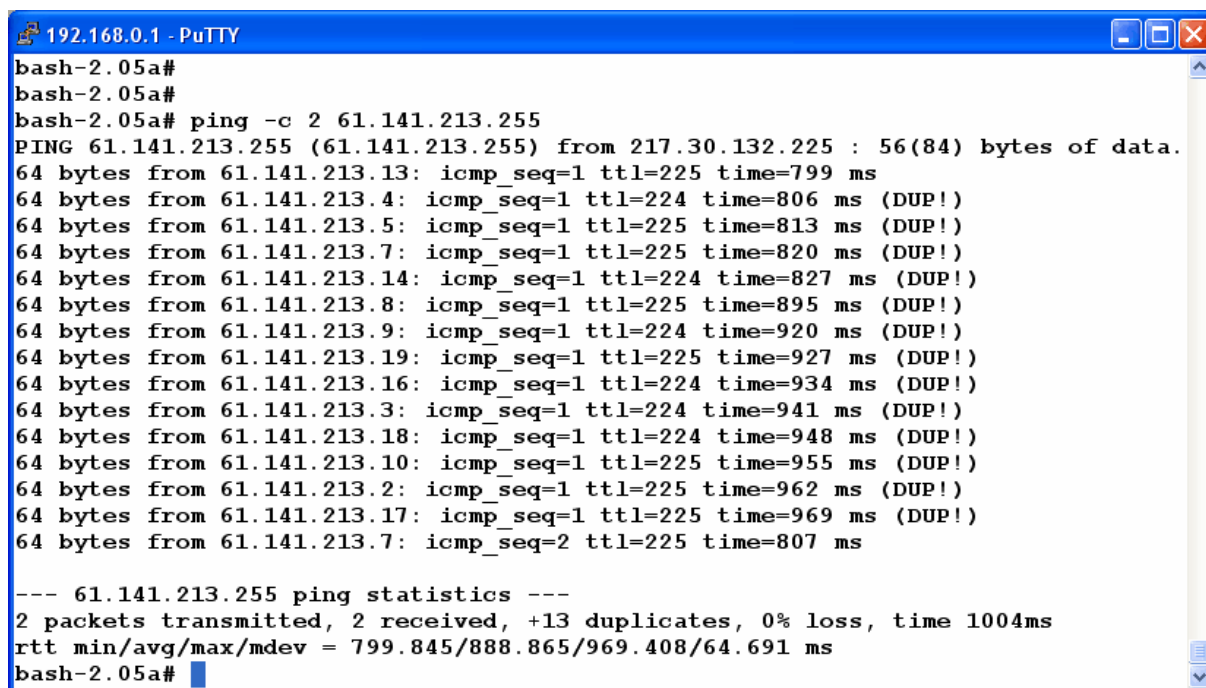
Podczas ataku typu SMURF wykorzystuje się pakiety Echo Żądanie/Odpowiedź protokołu ICMP. Przebieg ataku ma przeważnie następujący charakter

- Atakujący tworzy ping (ICMP echo request) i wysyła go na adres rozgłoszeniowy sieci, podając jednocześnie jako adres źródła IP ofiary
- Każdy host w sieci przygotowuje odpowiedź ICMP echo reply i odsyła go na adres IP ofiary



Rysunek 16 Przebieg ataku typu ICMP SMURF

Atak ten ma na celu wygenerowanie tak dużego ruchu na łączu ofiary, aby jej klienci/użytkownicy nie mogli skorzystać z jej usług. Warunkiem powodzenia ataku jest wysyłanie dużych (do 64 KB) pakietów z maksymalną szybkością na jaką pozwala łącze atakującego. Zwielokrotnienie siły ataku zależne jest w dużej mierze od ilości komputerów, które odpowiadają na adres rozgłoszeniowy (ang. broadcast). Napisanie automatu, który po kolei zeskakuje pewne pule adresów IP oraz wykryje wielokrotne odpowiedzi jest bardzo proste. Jako narzędzie dobrze nadaje się obecny we wszystkich systemach polecenie ping:



```
bash-2.05a#  
bash-2.05a#  
bash-2.05a# ping -c 2 61.141.213.255  
PING 61.141.213.255 (61.141.213.255) from 217.30.132.225 : 56(84) bytes of data.  
64 bytes from 61.141.213.13: icmp_seq=1 ttl=225 time=799 ms  
64 bytes from 61.141.213.4: icmp_seq=1 ttl=224 time=806 ms (DUP!)  
64 bytes from 61.141.213.5: icmp_seq=1 ttl=225 time=813 ms (DUP!)  
64 bytes from 61.141.213.7: icmp_seq=1 ttl=225 time=820 ms (DUP!)  
64 bytes from 61.141.213.14: icmp_seq=1 ttl=224 time=827 ms (DUP!)  
64 bytes from 61.141.213.8: icmp_seq=1 ttl=225 time=895 ms (DUP!)  
64 bytes from 61.141.213.9: icmp_seq=1 ttl=224 time=920 ms (DUP!)  
64 bytes from 61.141.213.19: icmp_seq=1 ttl=225 time=927 ms (DUP!)  
64 bytes from 61.141.213.16: icmp_seq=1 ttl=224 time=934 ms (DUP!)  
64 bytes from 61.141.213.3: icmp_seq=1 ttl=224 time=941 ms (DUP!)  
64 bytes from 61.141.213.18: icmp_seq=1 ttl=224 time=948 ms (DUP!)  
64 bytes from 61.141.213.10: icmp_seq=1 ttl=225 time=955 ms (DUP!)  
64 bytes from 61.141.213.2: icmp_seq=1 ttl=225 time=962 ms (DUP!)  
64 bytes from 61.141.213.17: icmp_seq=1 ttl=225 time=969 ms (DUP!)  
64 bytes from 61.141.213.7: icmp_seq=2 ttl=225 time=807 ms  
  
--- 61.141.213.255 ping statistics ---  
2 packets transmitted, 2 received, +13 duplicates, 0% loss, time 1004ms  
rtt min/avg/max/mdev = 799.845/888.865/969.408/64.691 ms  
bash-2.05a#
```

Rysunek 17 Przykład odebrania pakietów ICMP z adresu rozgłoszeniowego

Na powyższym zrzucie ekranowym przedstawione jest skutek wykonania komendy ping, z parametrem określającym liczbę pakietów do wysłania (2) oraz adresem sieci docelowej (61.141.213.255). W tym konkretnym przypadku w odpowiedzi przyszły 2 odpowiedzi oraz 16 duplikatów.

Wyszukiwanie dużej ilości takich sieci dla agresora byłoby czasochłonne, dlatego są w Internecie serwisy z gotowymi listami sieci, zarówno w formie do przeglądania, jak i w postaci plików tekstowych. Te ostatnie idealnie nadają się do zaimportowania w narzędziu agresora. Jednym z takich serwisów jest <http://www.netscan.org> [34]. Oficjalnym celem serwisu jest ostrzeganie administratorów źle skonfigurowanych sieci (serwis po pozytywnym przeskalowaniu wysyła list), jednak upublicznienie tych danych bardziej przydaje się chyba agresorom.

#### 6.3.1.1.2. Obrona

Ataki tego typu są możliwe z dwóch powodów. Oba z nich wiążą się z błędną konfiguracją ruterów i brakiem odpowiednich zabezpieczeń na najniższym poziomie - w sieciach firmowych, uczelnianych itp. Pierwszy z nich, brak reguł filtrujących fałszywe adresy nadawcy w pakietach opuszczających LAN pozwala na prowadzenie ataków smurf z danej sieci. Drugi, włączone directed broadcasts na routerze brzegowym, pozwala na wykorzystanie danej sieci jako wzmacniacza. Ponadto już w 1989 roku w dokumencie RFC 1122 dopuszczono możliwość, by żądania ICMP wysyłane na adres rozgłoszeniowy lub w trybie wielonadawania (multicast) były ignorowane. Do zalecenia tego stosują się systemy operacyjne z rodziny Windows. W systemie Linux, aby wyłączyć odpowiedzi na adres rozgłoszeniowy należy wydać polecenie:

```
echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

natomiast na routerach firmy Cisco należy dodać do konfiguracji opcje:

```
no ip directed-broadcast
```

Problem następuje, kiedy jesteśmy już atakowani tą metodą. Pierwszą rzeczą jaką należy zrobić jest zablokowanie na routerze przesyłania pakietów ICMP-Echo-Replay. To spowoduje, że pakiety ICMP nie będą zalewać naszej sieci wewnętrznej. Gorzej jest, jak całkowity ruch wygenerowany przez agresora jest większy niż łączną przepustowość naszych łączy. Przykładowo, jeśli by agresorowi udało się powielić ruch do 4 Mb/s, a nasze łącze posiada przepustowość ok. 2 Mb/s, to zablokowanie pakietów ICMP i tak nic nie da. Jedynym wyjściem jest kontakt z providerem, do którego jesteśmy podłączeni. Tylko on może w swojej sieci szkieletowej zablokować powódź pakietów oraz spróbować ustalić ich pochodzenie. Jednakże w polskich warunkach w przypadku dużych operatorów (np. TP S.A. – Polpak) uzyskanie takiej pomocy jest raczej niewykonalne.

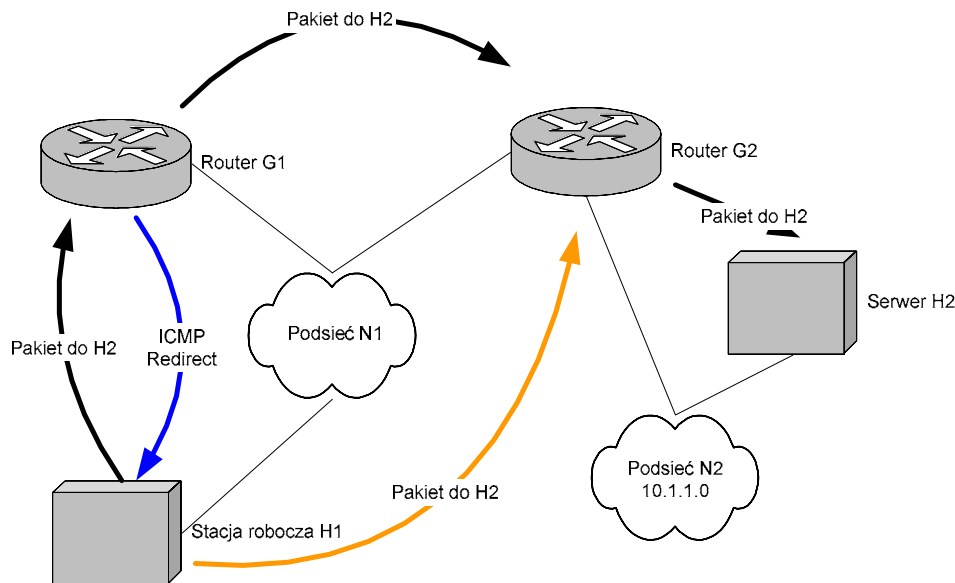
#### 6.3.1.2. ICMP – Redirect

Urządzenia podłączone do sieci pamiętają listę bram, poprzez które mają dostęp do urządzeń znajdujących się w innych podsieciach. Jedną spośród nich jest bramą domyślną. Jeśli dostępnych bram jest więcej niż jedna, mogą się zdarzać sytuacje, że pakiet zostanie wysłany do niewłaściwej. Przykład wymiany komunikatów w takim przypadku przedstawia rysunek. Bramą domyślną dla urządzenia H1 jest G1. Jeśli H1 nie posiada w swojej tablicy routingu wpisu dokąd kierować pakiety przeznaczone dla sieci 10.1.1.0/24, zostaną one wysłane poprzez G1. Brama G1 wykonuje następujące czynności:

- Sprawdza czy pakiet został wysłany przez urządzenie podłączone do tej samej podsieci co on
- Przeszukuje własną tablicę routingu i sprawdza czy urządzenie, przez które powinien przesłać pakiet (G2) oraz urządzenie, z którego on pochodzi (H1) znajdują się w tej samej podsieci

Jeśli obydwa warunki zostaną spełnione G1 wysyła komunikat ICMP Redirect do H1 z informacją, że właściwą bramą dla pakietu jest G2, natomiast do G2 jest wysyłany pakiet otrzymany wcześniej od H1.

Urządzenie H1, po odebraniu komunikatu ICMP Redirect powinno uaktualnić swoją tablicę routingu tak, aby kolejne dane były już wysyłane do G2.

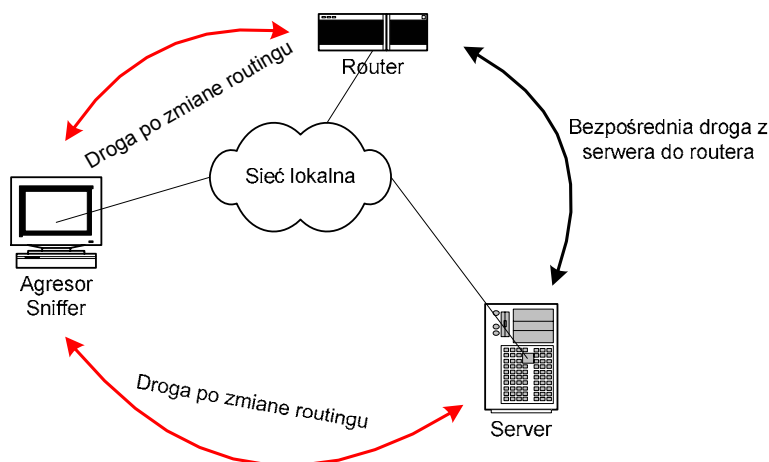


Rysunek 18 Działanie komunikatu ICMP Redirect

#### 6.3.1.2.1. Anatomia ataku

Pakiety ICMP Redirect można wykorzystać do ataku DoS na dwa sposoby. W 2001 odkryto błąd w routerach Cisco bezpośrednio związany z pakietami ICMP Redirect [35]. System operacyjny IOS, po otrzymaniu tego pakietu z nieznanym dla niego adresem (np. wygenerowanym losowo) zapamiętywał go w specjalnej tabeli. Problem polegał na tym, że w niektórych wersjach IOS'u wielkość tabeli nie miała limitu. To doprowadzało do wyczerpania pamięci operacyjnej routera. Skutkowało to tym, że niektóre urządzenia przestawały pełnić swoją funkcję, tzn. przysyłać pakiety oraz często niemożliwym było załogowanie się do konsoli routera. Więcej informacji można znaleźć w opisie błędu [35].

W drugim sposobie agresor może wysyłać pakiety ICMP Redirect do komputerów i routerów wymuszając zmianę trasy routowania pakietów. Wbrew intencją administratora pakiety mogą zacząć przechodzić przez komputer agresora umożliwiając mu podsłuchiwanie ruchu. Zdecydowana większość danych przesyłanych w Internecie jest nie kodowanych – hasła przesyłane czystym tekstem można odczytać z protokołów POP3, IMAP (do odbierania poczty), FTP, TELNET (często używany do urządzeń specjalnego przeznaczenia, jak serwery wydruku, proste routery, switchy konfigurowalne), HTTP (coraz więcej urządzeń sieciowych konfiguruje się poprzez przeglądarki WWW).



Rysunek 19 Przebieg ataku ICMP Redirect

#### 6.3.1.2.2. Obrona

Metoda obrony jest dość prosta - jeśli nasza topologia sieci na to pozwala (ustalony routing, bramy domyślne itp) należy powyłączać akceptacje komunikatów ICMP Redirect. Oczywiście blokada taka powinna znajdować także na bramie dostępowej, aby nie wpuszczać tego typu pakietów do sieci wewnętrznej.

Aby wyłączyć w systemie Linux wpływu pakietów ICMP redirect na routowanie pakietów, należy wydać polecenie

```
/bin/echo "0" > /proc/sys/net/ipv4/conf/all/accept_redirects
```

Natomiast w przypadku Windows XP/2000/2003 pomocne może być zmodyfikowanie następujących wartości rejestru. Nazwy wartości znajdują się w gałęzi drzewa:

HKLM\System\CurrentControlSet\Services\AFD\Parameters

Nazwa wartości: EnableICMPRedirect Rekomendowana wartość: 0

Poprawne wartości: 0 (wyłączone), 1 (włączone)

Opis: Wpisanie do wartości tego klucza liczby 0 powoduje, że routowanie pakietów się nie zmienia w przypadku otrzymania pakietu ICMP redirect.

#### 6.3.2. UDP

UDP [4] (ang. User Datagram Protocol - Datagramowy Protokół Użytkownika) to jeden z podstawowych protokołów internetowych. Umieszcza się go na czwartej warstwie (transportowej) modelu OSI. Jest to protokół bezpołączeniowy, więc nie ma narzutu na

nawiązywanie połączenia (inaczej niż w TCP). Nie ma też kontroli przepływu i wiarygodności. Protokół UDP sam w sobie nie zawiera żadnych mechanizmów auto-odpowiadania – sam w sobie nie może stać się powielaczem przy atakach typu DoS. Zagrożeniem są natomiast liczne aplikacje i serwisy działające na bazie tego protokołu. Zagrożenie stanowią takie usługi jak ECHO [36] (odpowiadając na pakiety odsyła otrzymane dane), CHARGEN [37] (generator strumienia znaków), czy DNS (rozproszona usługa zmiany nazw symbolicznych na numery IP i odwrotnie).

### 6.3.2.1. DNS

Posługiwanie tylko numerami IP w Internecie, jest dość uciążliwe ze względu na trudność w zapamiętaniu takiej postaci adresu przez użytkowników. Dlatego na ogół posługujemy się nazwami symbolicznymi takimi jak [www.onet.pl](http://www.onet.pl), poczt.wp.pl, [www.google.com](http://www.google.com) itd. Są one łatwiejsze do zapamiętania, gdyż są zbudowane z wyrazów, a nie cyfr jak w przypadku adresów IP. Mechanizm translacji nazw symbolicznych na adresy IP opiera się na 13 tzw. rootserwerach umieszczonych w strategicznych punktach Internetu. Rootserwery przechowują wpisy odpowiedzialne za funkcjonowanie podstawowych domen internetowych jak .com/.edu/gov/org itp. Każde państwo posiada także główne serwery DNS odpowiedzialne za przechowywanie informacji o domenach krajowych (np. pl/au/dk/uk).

#### 6.3.2.1.1. Anatomia ataku

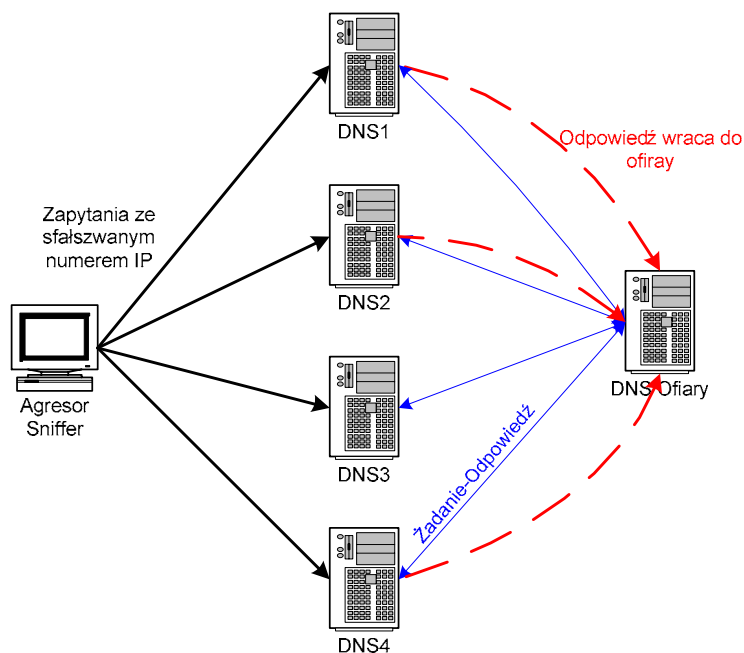
Serwery DNS oferują dwie możliwości powielenia. Pierwszą z nich jest proste wysłanie odpowiedzi na sfałszowane żądanie DNS. Żądanie w większości wypadków zawiera mniej danych niż odpowiedź na nie, dlatego można uzyskać znaczne wzmocnienie ruchu. Jednakże ta forma ataku, może być łatwo rozpoznana, ponieważ odpowiedzi będą przychodziły z portu źródłowego 53. Następnie można je wyfiltrować (jest to pewien jednak koszt). Wyfiltrowanie ruchu z portu 53 może również zablokować dostęp do DNS'ów ofiary przez inne zewnętrzne serwery uprawnione do tego. Można rozwiązać ten problem przez ustawienie wyjątków – zbioru serwerów uprawnionych.

Drugim sposobem jest wykorzystanie rekursywnego odpytywania serwerów DNS. Jeśli ofiara jest serwerem nazw dla konkretnych stref, agresor może wygenerować duży strumień zapytań do dużej ilości serwerów nazw. Te z kolei zaczną odpytywać serwer ofiary zmuszając ją do wysłania znacznej ilości tych samych danych. Zapytania do serwerów wysyłane przez agresora nawet nie muszą mieć sfałszowanego numeru źródłowego, co jest ważne w przypadku dobrej (poprawnej) konfiguracji sieci, w której znajduje się agresor. Jeśli sieć agresora nie zabrania mu wysyłania pakietów z sfałszowanym numerem IP, może to użyć adresu ofiary, powodując, że te same dane przejdą przez jego łącze dwukrotnie.

Przeszkodą w przeprowadzeniu takiego ataku jest mechanizm cachowania odpowiedzi. Po jednorazowym zapytaniu, każde kolejne nie powoduje już odwołań do głównego serwera,



lecz jest pobierane z pamięci. Atakujący może coraz to pytać o inne domeny/komputery i atak może się okazać skuteczny.



Rysunek 20 Przebieg ataku z wykorzystaniem serwerów DNS

#### 6.3.2.1.2. Obrona

Obrona przed atakami DoS wykorzystującymi DNS jest dość trudna. Powodzenie ataku zależy od znalezienia dostatecznej liczby serwerów DNS przez agresora, które umożliwiają zapytania rekursywne. Do tej pory większość serwerów DNS pozwala na ich wykonywanie, bez żadnych ograniczeń – były dostępne dla wszystkich. Ostatnio jednak obserwuje się większą dbałość o ich konfigurację i ograniczanie dostępu. W idealnym przypadku, każda podsieć ma swój serwer DNS, który przyjmuje zapytania rekursywne tylko od jej członków. Niestety w obecnej chwili można z łatwością znaleźć setki tysięcy pomocników (serwerów DNS) w ataku DoS.

Trudno również dobrać odpowiednią konfigurację zapory ogniowej. Żądania rekursywne w większości przypadków są wysyłane z portu 53 na port 53 protokołu UDP pomiędzy serwerami DNS.

#### 6.3.2.2. Chargen

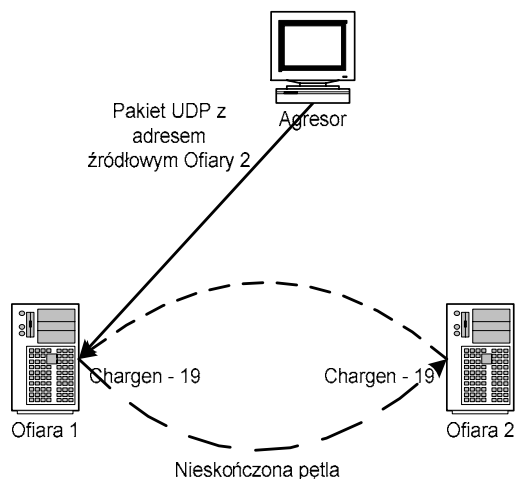
Chargen [37] jest skrótem od ang. Character Generator i jest to usługa generująca przypadkową ciąg znaków. Jej odmianą działającą na protokole UDP wysyła pakiet z losową długością od 0 do 512 znaków. Pakiet ten jest wysyłany w odpowiedzi na dowolny pakiet UDP, który przyjdzie na port 19.

W wersji TCP serwer wysyła nieprzerwanie strumień danych od momentu nawiązania połączenia, aż do jego zamknięcia. W zamierzeniach usługa miała ułatwić testowanie połączeń sieciowych i odnajdywanie powodów odrzucania pakietów.

#### 6.3.2.2.1. Anatomia ataku

Atak na usługę Chargen jest wyjątkowo prosty. W pierwszym wariantcie można po prostu zalewać serwer pakietami UDP o minimalnej długości. Zakładając, że serwer będzie odpowiadał pakietami o długościach od 0 do 512 znaków, można uzyskać kilkukrotne powielenie ruchu.

O wiele ciekawszy jest wariant, kiedy uda się znaleźć dwie działające usługi Chargen. Wystarczy wtedy wysłać do jednej z nich pakiet UDP ze sfalszowanym adresem źródłowym drugiej usługi i pakiety zaczną automatycznie krążyć w nieskończonej pętli. Wstrzyknięcie kilku milionów pakietów, spowoduje całkowity paraliż infrastruktury sieciowej u oby ofiar.



Rysunek 21 Przebieg ataku UDP Chargen

#### 6.3.2.2.2. Obrona

Jeśli któreś z urządzeń w naszej sieci ma włączoną usługę Chargen, należy ją bezwzględnie wyłączyć. W tej chwili większość systemów operacyjnych nie włącza tej usługi podczas instalacji domyślnej, dlatego bardzo trudno jest dla atakującego wykorzystać tą metodę. Dobrze jest również zablokować na zaporze ogniowej wejście i wyjście na port 19 TCP/UDP.

### 6.3.3. TCP

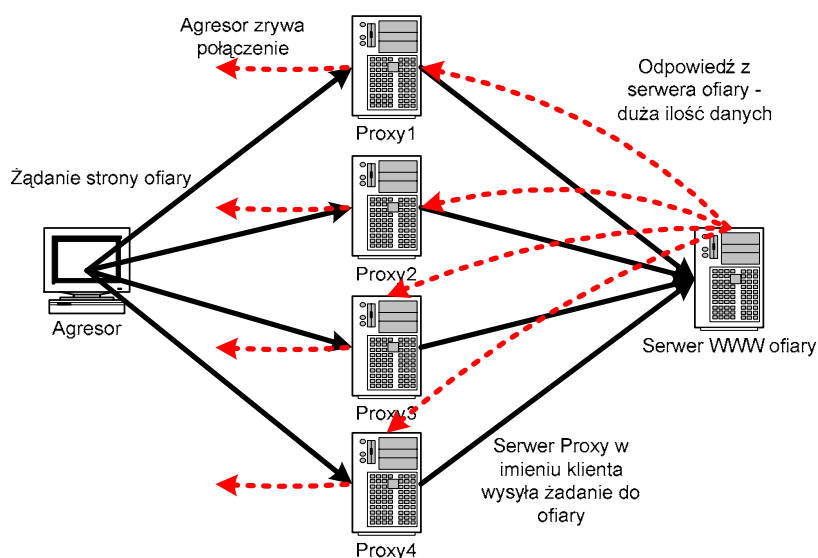
#### 6.3.3.1. HTTP

Protokół HTTP [42] stanowi najważniejszą część ogólnoswiatowej sieci World Wide Web. Działanie serwera opiera się na przyjmowaniu od klienta pojedynczych rozkazów i wykonywaniu ich. Każdy rozkaz wykonywany jest niezależnie od pozostałych. Właśnie dzięki protokołowi HTTP przesyła się żądania udostępniania dokumentów WWW i informacje o wybraniu odnośnika oraz informacje z formularzy.

Podczas kiedy typowej sesja protokołu HTTP następuje transfer danych pomiędzy serwerem a klientem, Proxy HTTP dostarcza mechanizm, dzięki któremu klient może wymusić na serwerze nawiązanie połączenia z serwerem ofiary. To teoretycznie może służyć do przeprowadzenia ataku DoS.

##### 6.3.3.1.1. Anatomia ataku

Aby przeprowadzenie ataku było możliwe, agresor musi posłużyć się, co najmniej kilkudziesięcioma serwerami HTTP Proxy. Następnie do wszystkich jednocześnie wysyła żądanie pobrania dużego zasoby (np. pliki multimedialne, długie dokumenty, archiwa). Znalezienie takich zasobów na stronach większości firm nie stanowi problemu. Serwery Proxy realizując żądanie agresora łączą się z ofiarą i pobierają dane. Jeśli serwery Proxy będą dysponowały wystarczającą przepustowością spowodują saturację łącza klienta i tym samym utrudnią prace uprawnionym użytkownikom. Agresor to pewnym interwale powinien rozłączyć się ze wszystkimi serwerami Proxy, aby żądane dane nie spowodowały saturacji jego z kolei łącza.



Rysunek 22 Przebieg ataku z wykorzystaniem HTTP Proxy

#### 6.3.3.1.2. Obrona

Przedstawiony powyżej scenariusz jest dość teoretyczny. Na drodze do przeprowadzenia go, stoi kilka trudności. Po pierwsze, aby móc zażądać strony, agresor musi dokonać trójstopniowego nawiązania połączenia. To oznacza, że nie może posłużyć się sfalszowanym adresem źródłowym. Następnie ten adres agresora nie tylko jest zapisywany w pliku logowania serwera Proxy, ale również w większości wypadków jest wysyłany do serwera ofiary w nagłówku żądania HTTP. Serwer ofiary może, a nawet w większości wypadków robi to domyślnie i zapisuje adres klienta, dla którego serwer Proxy wysłał żądanie. Znalezienie wystarczającej ilości serwerów Proxy, które są podłączone szybkimi łączami i zapewniają wystarczającą anonimowość jest bardzo trudne, co czyni ten rodzaj ataku trudnym do przeprowadzenia. Dodatkowy kłopot sprawia podstawa działania serwerów Proxy, a mianowicie buforowanie stron. Dany zasób będzie tylko raz pobierany z serwera ofiary, a każda kolejna próba jest żądania, spowoduje pobranie go z dysku Proxy.

### 6.4. Ping of Death

Atak Ping of Death [38] polega na wysłaniu do odległej maszyny pofragmentowanego datagramu ICMP Echo request o łącznej długości przekraczającej 65 535 bajtów, czyli maksymalną długość pakietu, jaką można wpisać do nagłówka IP. Niektóre systemy operacyjne nie były w stanie poprawnie przetworzyć takiego pakietu, co zwykle powodowało zawieszenie się lub restart maszyny.

#### 6.4.1. Anatomia ataku

Zgodnie z RFC-791 pakiet może mieć rozmiar do 65,535 ( $2^{16}-1$ ) bajtów długości, które musi zawierać nagłówek (zwykle 20 bajtów, jeśli nie zastosowano opcji IP). Pakiety, które są większe niż niższa warstwa protokołu potrafi przesłać (parametr MTU) są dzielone na mniejsze części a następnie składane w całość przez odbiorcę. Dla ethernetu MTU wynosi zwykle 1500 bajtów.

Pakiet ICMP Echo znajduje się wewnątrz pakietu IP i składa się z ośmiu bajtów nagłówka (RFC-792), po których występują dane. Dlatego maksymalny rozmiar danych wynosi:  $65535 - 20 - 8 = 65507$  bajtów.

Wysłanie większej ilości danych niż 65507 bajtów było możliwe ze względu na sposób fragmentacji pakietów. Fragmentacja opiera się na wartości przesunięcia (offset) w każdym fragmencie, w celu ustalenia gdzie ten pojedynczy pakiet ma się znaleźć podczas składania. Dlatego w ostatnim fragmencie, jest możliwe takie ustalenie przesunięcia w pasującym rozmiarze fragmentu, że przesunięcie plus rozmiar będzie większe niż 65535.

Systemy operacyjne nie przetwarzają pakietu, dopóki nie dokonają jego całego złożenia, dlatego było możliwe przepełnienie 16 bitowych rejestrów, co prowadziło do załamania systemu, restartów itp.

Do przeprowadzenia tego ataku nie potrzeba specjalnych narzędzi, wystarczy w linii poleceń Windows wpisać np.: `C:\ping -l 65540`

### 6.4.2. Obrona

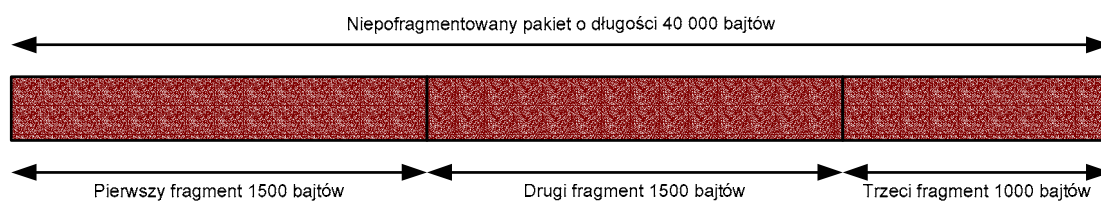
Na ten rodzaj ataku, który pojawił się w roku 1996 było podatnych wiele systemów operacyjnych oraz urządzeń sieciowych. W tej chwili ten rodzaj ataku nie stanowi żadnego zagrożenia.

## 6.5. Teardrop

Około roku 1996/1997 odkryto wiele błędów w implementacjach stosów TCP/IP. Po błędach umożliwiających atak DoS typu LAND i Ping of Death, pojawił się kolejny rodzaj ataku nazwany Teardrop [47] wykorzystujący lukę w składaniu pofragmentowanych pakietów. Wysyłanie specjalnie utworzonych pakietów powodowało bądź restart maszyny, jej blokadę, bądź na tyle spowolnienie jej, że uprawnieni użytkownicy nie mogli korzystać z jej zasobów.

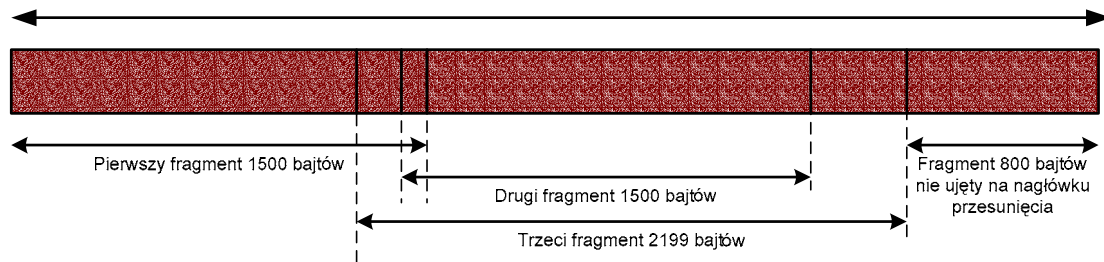
### 6.5.1. Anatomia ataku

Trzy części prawidłowo pofragmentowanego pakietu o długości 4000 bajtów, powinny w polu znacznika przesunięcia zawierać wartości: np. 1 do 1500, 1501 do 3000 i 3001 do 4000.



Rysunek 23 Prawidłowo pofragmentowany pakiet

Teardrop tak ustawia te wartości, że kolejne kawałki pakietu zachodzą na siebie - np. 1 o 1500, 1500 do 3000 i 1001 do 3200. Co więcej - cały pakiet nie zostanie odebrany, bowiem w ostatni kawałek wpisano wartość (3200) niższą od długości pierwotnego pakietu. Host odbierający nie jest w stanie poprawnie złożyć pakietu, co może prowadzić do większego obciążenia procesora (obsługa wielu błędów rekonstrukcji) lub nawet restartu maszyny. Linux w wersjach jądra starszych od 2.0.32 nie radził sobie z pakietami, w których jeden kawałek całkowicie zawierał się w innym. Jedno z narzędzi do wykonywania ataków typu DoS wysyłało następujący pakiet:



Rysunek 24 Nieprawidłowy pakiet w ataku Teardrop

### 6.5.2. Obrona

Również jak pozostałe ataki typu DoS (Ping of Death oraz LAND) dzisiaj większość urządzeń i systemów operacyjnych jest całkowicie odporna na tego typu atak.

## 7. Narzędzia atakującego

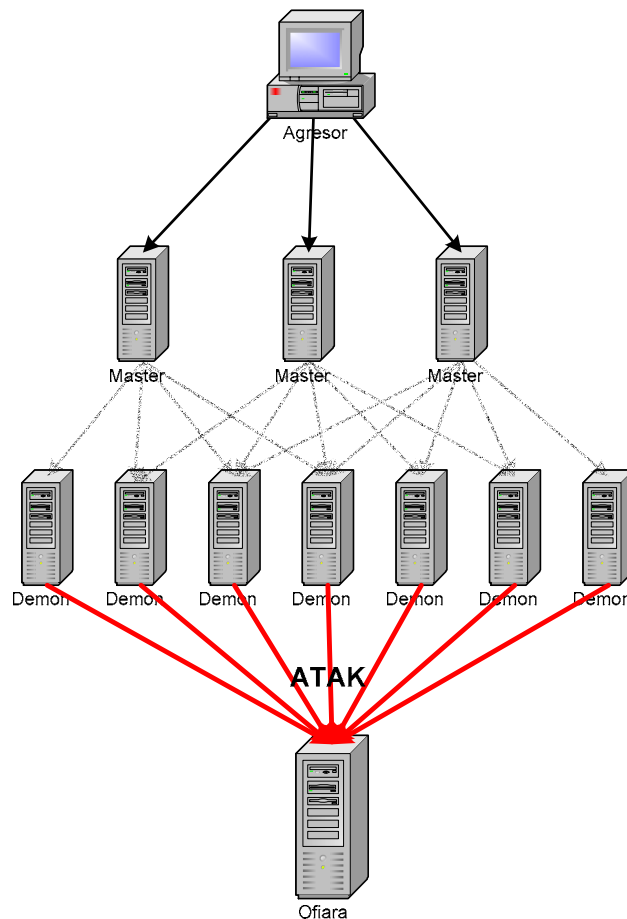
Tendencje w rozwoju ataków DDoS przesuwają się w kierunku autonomicznych agentów, którzy w żaden sposób nie komunikują się z sobą. Agenci mają w swoim kodzie już z góry określony scenariusz ataku, czyli: czas, długość jego trwania, typ ataku oraz oczywiście ofiarę. Zapewnia to dużą anonimowość i bezpieczeństwo dla agresora, gdyż rzadko udaje się ustalić pierwszą infekcję. Jednak na początku rozwoju tego typu ataków były one przeprowadzane bądź ręcznie lub półautomatycznie. W obu tych rodzajach sama komenda ataku była bezpośrednio wydawana przez agresora. W roku 2000 pojawiło się kilka narzędzi, które umożliwiały budowanie sieci DDoS i przeprowadzanie ataków. Najbardziej znane z nich to: Trinoo, TFN oraz Stacheldraht.

### 7.1. Trinoo

Oryginalne demony Trinoo [39] zostały znalezione w postaci binarnej na wielu maszynach z systemem Solaris 2.x oraz Linux RedHat 6.0. Włamanie do tych systemów było możliwe dzięki błędowi przepełnienia bufora w usługach RPC (statd, cmsd, ttdbserverd).

Na początku sądzono, że Trinoo jest kolejnym typem tylnej furtki z możliwością wysyłania komend przez protokół UDP oraz mechanizmem podsłuchiwania sieci. Później odkryto główny cel tego narzędzia jakim był atak DDoS. W roku 1999 sieć Trinoo składała się prawdopodobnie z setek, może nawet tysięcy maszyn. Część z nich (około 227 maszyn oraz 114 w sieci Internet2) rozpoczęły 17 sierpnia 1999 roku powódź na jeden z serwerów Uniwersytetu w Minnesocie, powodując paraliż sieci przez prawie dwa dni.

Budowa sieci DDoS, na jaką umożliwiał pakiet Trinoo była wielowarstwowa:



Rysunek 25 Schemat typowej sieci DDoS (Trinoo)

Agresor lub agresorzy komunikowali się z głównymi serwerami (master). Następnie każdy z tych serwerów kontrolował wiele demonów, czyli maszyn bezpośrednio generujących powódź pakietów.

Wszystko, co agresor musiał zrobić, aby rozpocząć atak było zalogowanie się do głównych serwerów, podanie hasła i komendy do ataku. Do komunikacji pomiędzy poszczególnymi warstwami były używane porty:

- Agresor do Mastera: 27655/tcp
- Master do Demona: 27444/udp
- Demon do Master: 31335/udp

Demony atakowały ofiarę siłowo zalewając ją pakietami UDP lub TCP.



## 7.2. TFN (Tribe Flood Network)

Tribe Flood Network [40] został napisany przez Mixtera. W TFN klient jest odpowiednikiem mastera z Trinoo. TFN umożliwia przeprowadzanie wielu rodzajów ataków:

- Powódź ICMP
- Powódź pakietów SYN
- Powódź pakietów UDP
- Ataki typu SMURF

TFN ma możliwość generowania ruchu ze sfałszowanym numerem IP. Umożliwia również uruchomienie powłoki z prawami root'a na porcie TCP. Komunikacja pomiędzy klientem a deamonem odbywa się za pomocą ICMP ECHO REPLAY, co znacznie utrudnia lub wręcz uniemożliwia blokowanie ataku za pomocą rozwiązań typu firewall. Komendy są wysyłane za pomocą 16-bitowej liczby umieszczanej w polu id pakietu, sequence number ma zawsze wartość 0. Słabością tego narzędzia (ze strony ofiary – zaletą) jest brak autoryzacji przy wysyłaniu komend. Jeśli domyślne numery komend nie zostały zmienione, wystarczy wysłanie jednego pakietu, aby wyłączyć demona (powstał skrypt napisany w języku perl o nazwie 'civilize').

## 7.3. Stacheldraht

Stacheldraht [48], co znaczy po niemiecku drut kolczasty, jest następcą pakietu TFN rozszerzając go o kilka możliwości. Pierwszą z nich jest kodowana transmisja wewnątrz sieci DDoS pomiędzy agentami a demonami, co utrudnia dość dobrze wykrycie jej przez systemy IDS. Drugą nowością jest możliwość wymuszenia na agentach aktualizacji swojego kodu poprzez pobranie go z podanego w komendzie serwera. Pobranie plików następuje poprzez protokół RPC i mogą być do tego wykorzystane inne konta zdobyte przez włamywacza.

Narzędzie umożliwia budowanie sieci DDoS podobnej do tej z TFN, w której są komputery główne (master) i podrzędne (demony). Atakujący musi włamać się odpowiednią ilość maszyn, zainstalować pakiet oraz dodatkowo tzw. root-kit, aby ukryć w systemie obecność dodatkowych usług. Zainstalowanie root-kita jest szczególnie ważne dla masterów, ponieważ ich rola w sieci DDoS jest kluczowa. Etap ten był częściowo automatyzowany poprzez odpowiednie skrypty i było możliwe zarażenie kilkuset w krótkim czasie.

Kiedy sieć DDoS była już gotowa, atakujący używał narzędzie podobnego do telnetu, ale kodującego transmisje i łącząc się do masterów wydawał im komendy. Te z kolei kontaktowały się z demonami również szyfrując dane. Podobne jak TFN Stacheldraht umożliwiał atakowanie poprzez powódź UDP, ICMP, SYN Flood oraz SMURF.

## 8. Projekt i implementacja systemu wykrywania ataków DOS/DDOS – BIDS

BIDS (Bayesian Intrusion Detection System), jest aplikacją, której celem jest wykrywanie ataków typu DoS/DDoS. BIDS został w całości napisany przez autora tej pracy praktycznie od zera (pomijając zastosowane biblioteki). Autor nie opierał się na żadnym innym działającym systemie tego typu. Aplikacja ta koncentruje się na wykrywaniu anomalii w natężeniu ruchu. Szczególnie czuła jest na zwiększony ruch w wybranych protokołach. Takie założenie projektowe spowodowało, że system BIDS nie wykryje ataku DoS/DDoS, który polega na wysłaniu jednego, specjalnie przygotowanego pakietu. W takich przypadkach bardzo dobrze sprawują się tradycyjne systemy IDS w rodzaju Snort [57], które posiadają swoją bazę sygnatur. BIDS nie jest konkurencją dla nich, tylko uzupełnieniem ich funkcjonalności.

Założeniem projektowym było nie tylko wykrywanie kilku popularnych ataków DoS/DDoS, ale także ostrzeganie o wzroście podejrzanego ruchu, który nie został zidentyfikowany. Wybrane charakterystyczne wykrywane ataki to:

- Powódź pakietów ICMP Redirect
- Atak typu ICMP Smurf
- Powódź pakietów ICMP Unreach
- Powódź UDP Chargen
- Atak na serwer DNS
- Atak typu SYN Flood
- Atak na serwer SMTP
- Atak na serwer WWW

Inne ataki, które nie pojawiły się na powyższej liście, zostaną zakwalifikowane jako:

- Powódź pakietów ICMP
- Powódź pakietów UDP
- Powódź pakietów TCP

System BIDS nie tylko wykrywa ataki, które są dokonywane na sieć chronioną, ale również ostrzega on o anomaliach ruchu wychodzącego z tej sieci. Przez to możemy zostać poinformowani, że pośredniczymy, na przykład, w ataku typu SMURF.

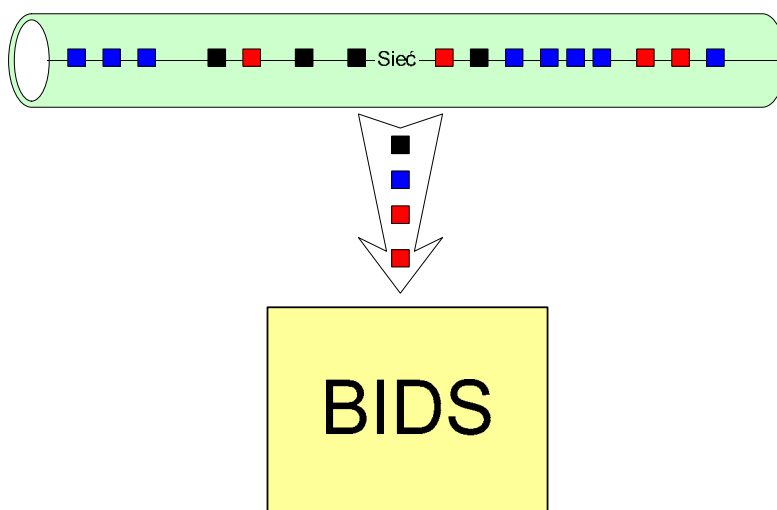
W kolejnych rozdziałach zostanie dokładnie opisany proces wyboru miejsca implementacji, wymagania, proces kompilacji i instalację oraz kolejnych faz uruchamiania. Dla osób

zainteresowanych wewnętrznymi mechanizmami, nieodzowny będzie opis techniczny zawierający opis poszczególnych modułów, diagramy przepływu danych oraz ważniejsze algorytmy i rozwiązania.

## 8.1. Koncepcja działania systemu

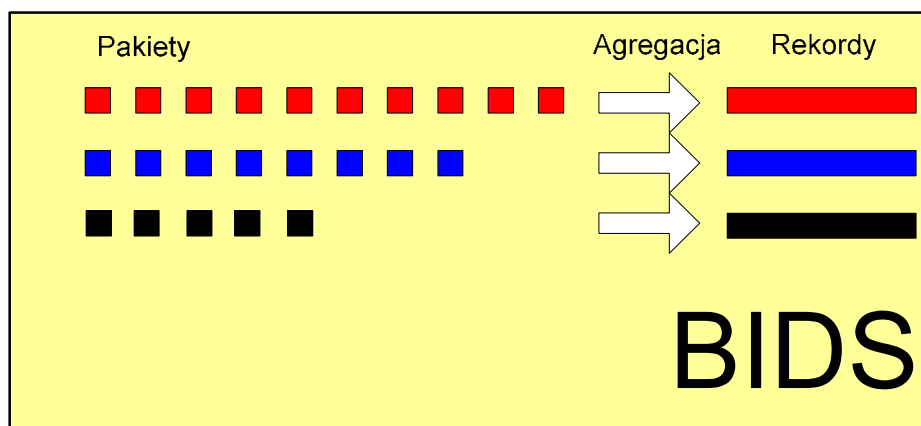
W tym rozdziale zostanie przedstawiona ogólna koncepcja działania systemu, bez zbędnych szczegółów technicznych. Rozwinięciem tego rozdziału jest opis techniczny. (8.3)

Aplikacja BIDS przełącza interfejs sieciowy w tryb nasłuchiwania pełnego (ang. promiscuous). W tym trybie karta sieciowa potrafi odbierać wszystkie pakiety, które fizycznie do niej docierają, mimo, że nie są przeznaczone dla niej (inny docelowy adres MAC). System BIDS umożliwia zdefiniowanie przez użytkownika dodatkowych filtrów, które ograniczą ruch brany pod uwagę przez samą aplikację.



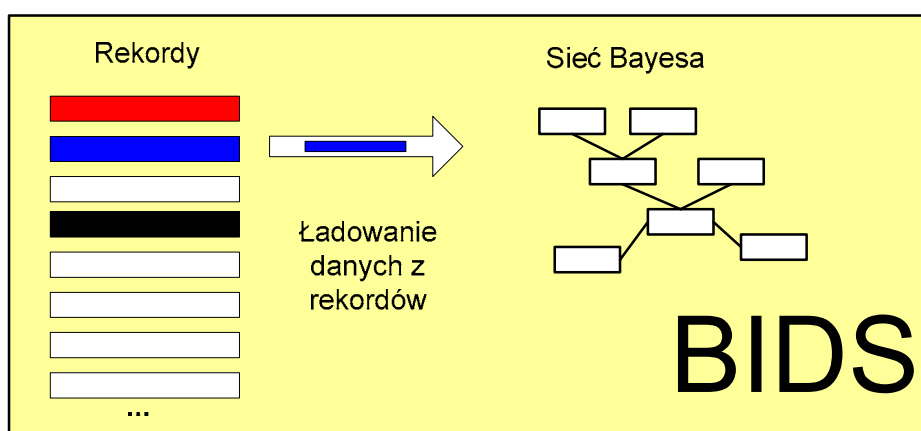
Rysunek 26 System BIDS – przechwytywanie pakietów

Przechwycone pakiety są klasyfikowane na dwie kategorie: ruch do chronionej sieci i ruch od chronionej sieci. Pozostałe pakiety są odrzucane. W następnym kroku dane z docierających pakietów (np. rozmiar, ilość) są sumowane z innymi. W ten sposób następuje agregacja danych. Zagregowane dane są następnie zapisywane do pamięci operacyjnej i czekają na zanalizowanie. Dane te tworzą rekordy. Rekord może odpowiadać pojedynczemu połączeniu (adres źródłowy IP, port źródłowy, adres docelowy IP, port docelowy), jak i może przedstawiać ruch na daną usługę w sieci chronionej (wszystkie połączenia do serwera WWW).



Rysunek 27 System BIDS - agregacja danych z otrzymanych pakietów

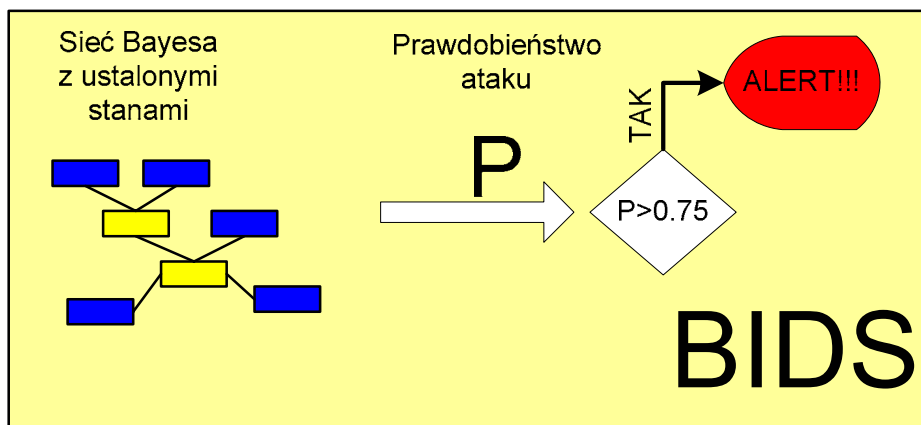
Każdy rekord jest analizowany cyklicznie, co jakiś czas. Interwał ten jest zależny od aktualnego obciążenia systemu, jednak nie jest mniejszy od 4 s (domyślnie). Następnie system wyszukuje w pamięci te rekordy, które zostały zmodyfikowane (przyszły pasujące pakiety) od ostatniej analizy. Jeśli rekord spełnia te warunki, to dane z niego są przepisywane do węzłów sieci Bayesa – wartości oznaczają stan węzła. Stany wszystkich węzłów mają charakter dyskretny. Dla niektórych wartości, jak ilość pakietów na sekundę system stosuje specjalne funkcje dyskretyzujące.



Rysunek 28 System BIDS – ładowanie danych z rekordów do sieci Bayesa

W dalszej kolejności sieć Bayesa oblicza prawdopodobieństwa nieustalonych węzłów. Jednym z nich (centralnym) jest węzeł oznaczający typ ataku. W ten sposób zostaje obliczone prawdopodobieństwo ataku  $P$ . W przypadku, gdy jest większe od zdefiniowanego progu (0.75 - domyślnie), na ekran jest wypisywane ostrzeżenie. Przykładowy alert ma postać:

```
Atak 'tcp_syn_flood' z prawdopodobienstwem 0.987 z 192.168.10.10:0 ->
192.168.0.1:70
```



Rysunek 29 System BIDS – obliczanie prawdopodobieństwa ataku

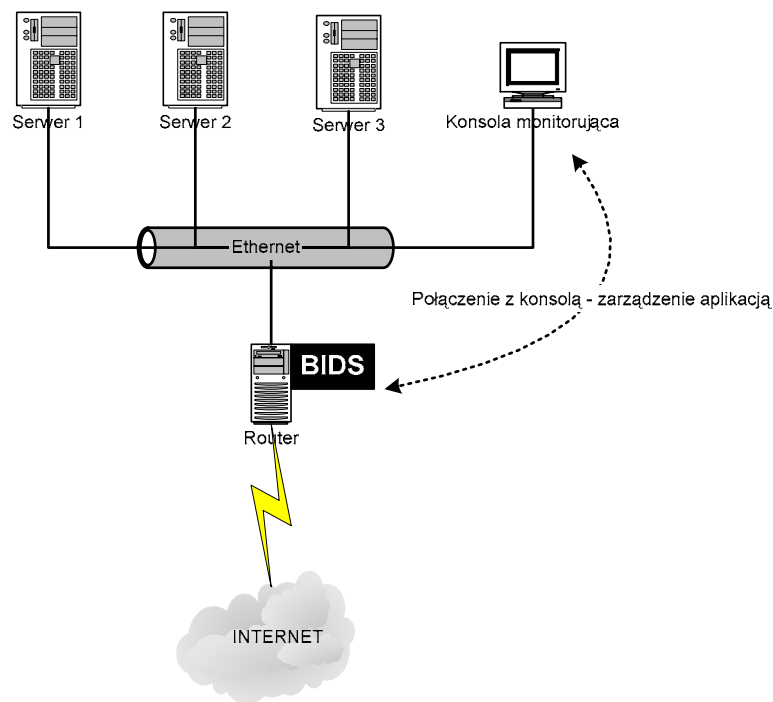
Każdy alert zawiera nazwę rozpoznanego ataku, jego prawdopodobieństwo, adres IP źródłowy, port źródłowy, adres IP docelowy, port docelowy.

Kiedy dany rekord został już zanalizowany, jego wartości są zerowane (ilość pakietów, sumaryczny rozmiar itp). Jeśli przyjdą kolejne pakiety, które będą pasować do danego rekordu, to dane o nich zostaną do niego zapisane. System BIDS dba o optymalne wykorzystanie pamięci, dlatego rekordy, które nie były przez dłuższy czas (60 s domyślnie) używane, są usuwane z pamięci.

## 8.2. Opis użytkowy

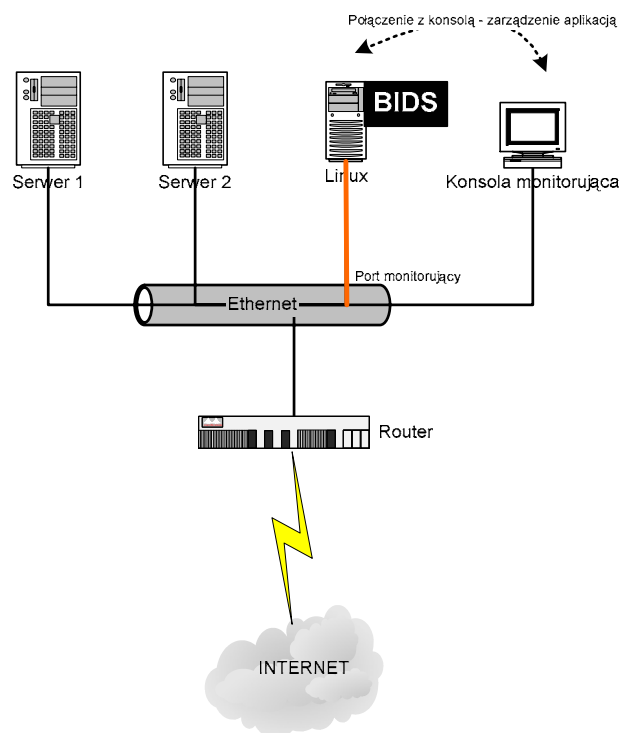
### 8.2.1. Miejsce instalacji systemu

System BIDS ma charakter pasywny. Obserwuje on tylko przechodzący ruch po sieci i nie wpływa na niego. Dobrym miejscem instalacji systemu jest brama sieciowa, która pośredniczy w przekazywaniu pakietów z sieci lokalnej do Internetu i odwrotnie. Instalacja jest możliwa o ile bramą jest komputer klasy PC z zainstalowanym systemem Linux.



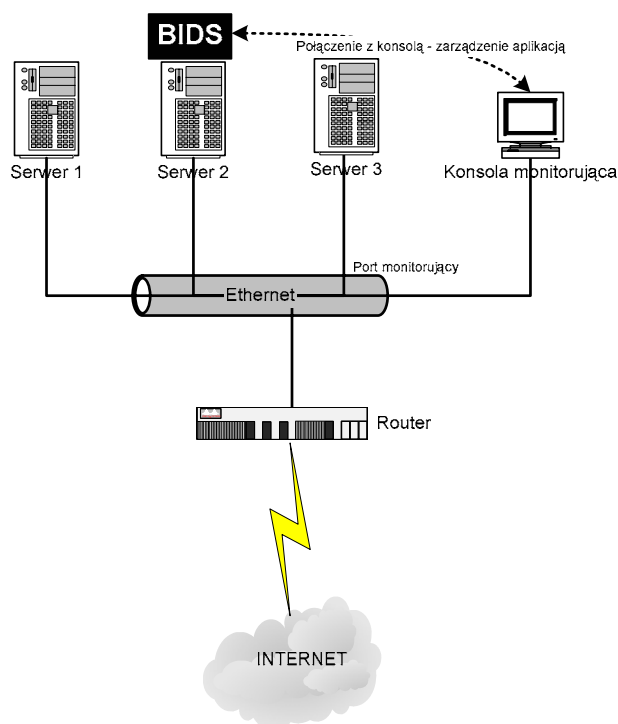
Rysunek 30 Instalacja BIDS na bramie sieciowej

W przypadku, kiedy bramą sieciową jest wyspecjalizowany router (np. Cisco, Intel), należy rozważyć instalację systemu wewnątrz chronionej sieci. System do swojego działania musi obserwować cały ruch wchodzący jak i wychodzący z chronionej sieci. Dziś standardem sieci lokalnych jest technologia Ethernet i przełączniki sieciowe (ang. switch). Dużą zaletą tych ostatnich jest to, że do danego portu w przełączniku jest wysyłany tylko ruch przeznaczony dla danego adresu MAC karty sieciowej. Ma to ochronić sieć przed podsłuchiwaniem ruchu. Jest to oczywisty kłopot przy wdrażaniu systemu BIDS. Dlatego w takiej sieci maszyna z zainstalowaną aplikacją musi być wpięta do tzw. portu monitorującego, na który będzie kierowany cały ruch.



**Rysunek 31 Instalacja BIDS wewnątrz sieci lokalnej**

Jeśli pierwsze dwa rozwiązania nie mogą być zastosowane, gdyż na przykład nie dysponujemy przełącznikiem z portem monitorującym, pozostaje trzecie rozwiązanie. Jest nim uruchomienie aplikacji tylko na jednym z serwerów sieci lokalnej. System BIDS będzie widział tylko ruch skierowany do tego komputera i będzie ostrzegał tylko o atakach DoS/DDoS skierowanych do tej maszyny. Jest to poważne ograniczenie, gdyż agresor może zaatakować inną maszynę, albo całą sieć.



Rysunek 32 Instalacja BIDS na jednym z serwerów

### 8.2.2. Wymagania i instalacja aplikacji

Aplikacja nie ma specjalnych wymagać sprzętowych. Podczas implementacji i testowania zupełnie wystarczał komputer klasy PC (Intel Pentium II 400 MHz, 64 MB RAM). Po wykonaniu rekompilacji nie powinno być problemów z uruchomieniem jej na innych architekturach procesorów (nie zostało to przetestowane).

Jeśli chodzi o wymagania programowe, aplikacja została napisana i przetestowana na systemie Linux (Linux RedHat 9 i Fedra Core1). Z powodzeniem można ją uruchomić na innych dystrybucjach o ile zostaną zainstalowane odpowiednie pakiety. Wymagane biblioteki to:

- libpcap (np. z pakietu libpcap-0.7.2-7.9.1.i386.rpm)
- libpthread (np. z pakietu glibc-2.3.2-27.9.i386.rpm)
- libm (np. z pakietu glibc-2.3.2-27.9.i386.rpm)
- libstdc++ 2.96 (np. z pakietu compat-libstdc++-7.3-2.96.118.i386.rpm)

Prawie wszystkie te biblioteki są domyślnie instalowane w systemie. W nowszych dystrybucjach jest jedynie problem z ostatnią z nich (libstdc++). Ze względu na to, że program wymaga jej wcześniejszej wersji, należy zainstalować dodatkowy pakiet z serii compat.



Aby skompilować program, oprócz standardowych pakietów służących do programowania (cpp, gcc, devel, make) wymagane są:

- compat-gcc-7.3-2.96.118.i386.rpm
- compat-libstdc++-devel-7.3-2.96.118.i386.rpm
- compat-gcc-c++-7.3-2.96.118.i386.rpm
- libnet10-1.0.2a-0.fdr.5.rh90.i386.rpm

Rozpakować archiwum ze źródłami można poleceniem

```
tar -zxvf bids.tar.gz
cd bids/
```

Przed kompilacją można sprawdzić, czy ustawienia domyślne w pliku config.h odpowiadają naszym wymaganiom. Dokładny ich opis można znaleźć w rozdziale (8.3.2.2). Następnie możemy skompilować aplikację poleceniem:

```
make
```

Kompilacja powinna zakończyć się bez błędów. Brak którejkolwiek z bibliotek lub plików nagłówkowych będzie prawdopodobną przyczyną błędu.

### 8.2.3. Parametry systemu BIDS

Uruchomienie aplikacji bez parametrów spowoduje wyświetlenie ich listy:

```
[root@localhost PD]# ./bids
Wersja: 1.0
Uzycie: bids -n adres_sieci -m maska_sieci -t -c -d [-a -i interfejs] filtr
```

Dozwolone są następujące parametry:

Tabela 4 Opcje linii poleceń systemu BIDS

Parametr	Opis	Przykład
-n adres_sieci	Aplikacja do swojego działania potrzebuje wiedzieć, połączenia, do jakiej sieci powinna monitorować. Parametr ten określa adres sieciowy chronionej sieci.	-n 192.168.0.0

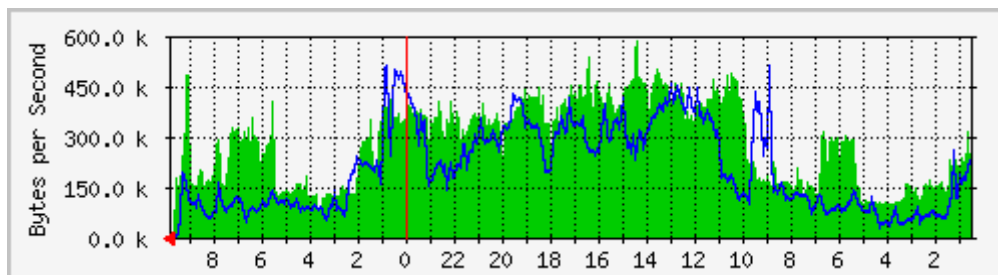
-m maska_sieci	Podobnie jak adres sieci, wymaganym parametrem jest maska sieci. Jej bitowa maska pozwala na określenie lokalnej części adresu IP maszyny, do której będzie przesyłany ruch.	-m 255.255.255.0
-t	Tryb badania statystyki sieci. Aplikacja w tym trybie tworzy statystyki przesyłanych pakietów i wypisuje je na standardowe wyjście.	-t
-c	Zaawansowany tryb, w którym program drukuje na standardowe wyjście plik służący do nauki sieci Bayesa (tzw. case file). Należy go stosować tylko wtedy, kiedy chcemy nauczyć sieć Bayesa nowych ataków, czy zmienić układ prawdopodobieństw w istniejącej. Z trybem tym powiązana jest opcja [-a], która powoduje, że generowane dane są oznaczone jako ataki.	-c
-d	Tryb wykrywania ataków typu DoS/DDoS. Jest to domyślny tryb pracy.	-d
-a	Opcja działa tylko w powiązaniu z parametrem [-c] i powoduje, że generowany plik nauki sieci Bayesa zawiera dane oznaczone jako ataki.	-a
-i	Interfejs sieciowy, na którym aplikacja ma nasłuchiwać. Nie podanie tego parametru spowoduje wybranie domyślnego interfejsu.	-i eth0
filtr	Dodatkowy filtr, który może zawęzić ruch brany pod uwagę przez aplikację. Aplikacja akceptuje bardzo rozbudowane możliwości budowania wyrażeń logicznych. Szczegółowy opis wszystkich opcji i przykłady można znaleźć na stronie projektu TCPDUMP. [49]	not host 192.168.0.10 not port 80

#### 8.2.4. Obserwacja parametrów sieci – dane wejściowe dla konfiguracji systemu

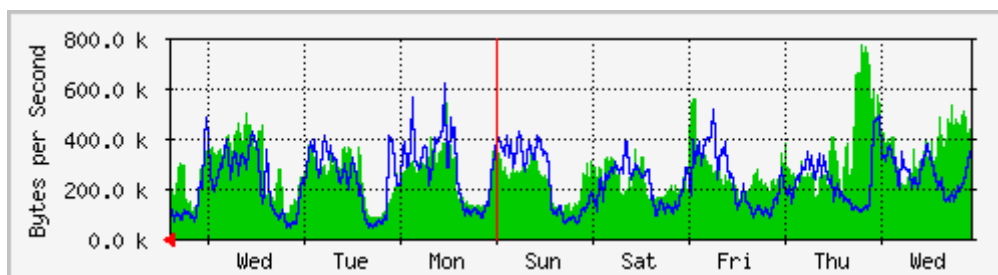
W każdej rzeczywistej sieci ilość, wielkość oraz natężenie przesyłanych danych jest inne. Dodatkowo wartości te zmieniają się w czasie na przestrzeni doby (duży ruch w dzień, mały w nocy), oraz w dłuższym okresie (zwiększanie ilości klientów – powolny wzrost ruchu). Parametry te zależą od bardzo wielu czynników, z których główne to:

- Liczba serwerów w chronionej sieci i ich konfiguracji
- Konfiguracja sieci (zapór ogniowych, filtrów i ograniczników ruchu)
- Rodzaj usług udostępnianych (protokoły, wielkości przesyłanych danych)
- Przepustowość łącz danych do dostawców ISP
- Liczba klientów i ich zachowania
- Kultura pracy w firmie (godziny pracy)

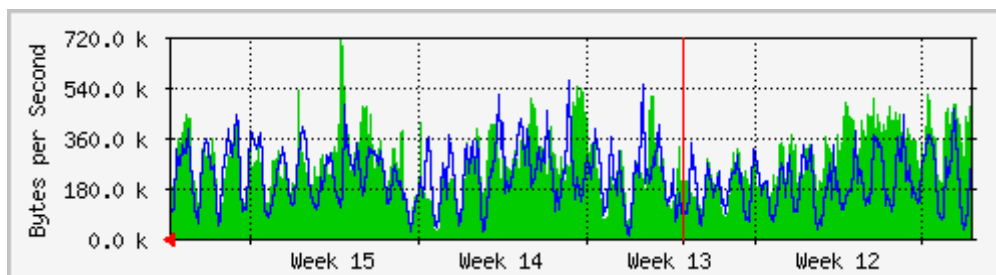
Przykładowe wykresy obciążenia łącz zostały pobrane od jednego z dostawców dostępu do sieci Internet (ISP).



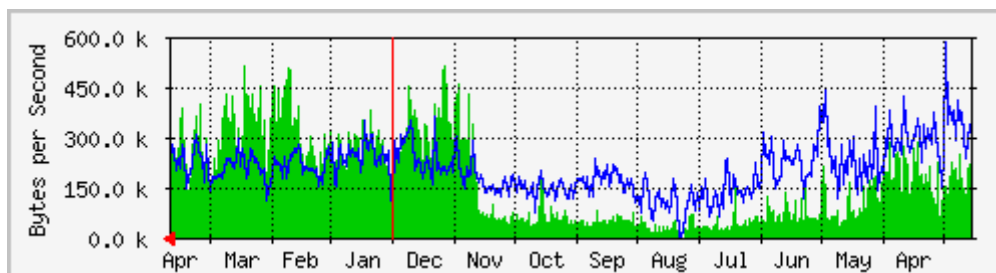
Rysunek 33 Przykładowy wykres ilości odebranych i wysłanych danych (okres doby)



Rysunek 34 Przykładowy wykres ilości odebranych i wysłanych danych (okres tygodnia)



Rysunek 35 Przykładowy wykres ilości odebranych i wysłanych danych (okres miesiąca)



Rysunek 36 Przykładowy wykres ilości odebranych i wysłanych danych (okres roku)

System BIDS potrzebuje do pracy średnich wartości przesyłanych danych. Wyznaczenie tego w sposób automatyczny jest zadaniem trudnym i może okazać się, że otrzymane wyniki będą błędne. Dodatkowo podczas automatycznej obserwacji sieci może zdarzyć się wiele zaburzeń, które zniekształcą wyliczone dane:

- Atak typu DoS/DDoS (nagły wzrost ruchu)
- Awaria łącza (prawie całkowity zanik ruchu)
- Prace administracyjne (wykonywana kopia zapasowa na inny serwer, rekonfiguracja sieci itp)

Z tego względu system BIDS wymaga od administratora ręcznego uruchomienia w specjalnym trybie, jak i również ręcznego zatrzymania. Podczas tego okresu będą mierzone średnie parametry przesyłanych danych. Takie działanie ma kilka zalet. Po pierwsze administrator może wybrać okres mierzenia parametrów (kilka godzin, doba, tydzień). Zmierzenie średniego ruchu w dzień, spowoduje, że system będzie prawdopodobnie za mało czuły w nocy (mniejszy ruch), natomiast uruchomienie tego trybu w nocy może spowodować, że w dzień jego czułość będzie za duża. To pozwoli administratorowi dobrać optymalny okres (np. doby). Drugim ważnym plusem jest to, że w przypadku niespodziewanego zdarzenia, które spowoduje zmniejszenie, lub zwiększenie ruchu pomiar może być przerwany i powtórzony jeszcze raz. Dlatego podczas tego trybu pracy zaleca się dodatkowo obserwację obciążenia sieci, na przykład przy pomocy takich narzędzi jak MRTG.

Aby uruchomić system w trybie obserwacji sieci, należy użyć parametru [-t]:

```
./bids -n 192.168.0.0 -m 255.255.255.0 -t net 192.168.0.0 > net.stat
```

Domyślnie aplikacja będzie generowała dane na standardowe wyjście, które zostało przekierowane do pliku `net.stat`. Plik ten będzie później przetwarzany. Opis formatu generowanych danych można znaleźć w rozdziale (8.3.2.3)

Następnie, wygenerowany plik należy przetworzyć przy pomocy skryptu PERL:

```
./net_stat_update net.stat
```

Uruchomienie skryptu spowoduje utworzenie (lub nadpisanie) pliku `net_stat.conf`. Dane zapisane do pliku, zostaną również wydrukowane na ekran. Dokładny opis pliku znajduje się w rozdziale (8.3.2.1). Podczas testów systemu zauważono, że wielkości dotyczące protokołów UDP i ICMP są często zbyt małe, co skutkuje za dużą czułością. Zaleca się, albo zasymulowanie małego ruchu ICMP/UDP podczas obserwacji sieci, lub poprawienie wartości ręcznie w pliku `net_stat.conf`. Stwierdzono, że przy średniej wielkości sieci wartości te powinny być ustawione na minimum:

- Liczba przesłanych pakietów na sekundę: 10
- Liczba przesłanych KB na sekundę: 10

### 8.2.5. Działanie systemu

Po utworzeniu pliku z statystyką sieci, można uruchomić system BIDS w docelowym trybie wykrywania ataków typu DOS/DDOS. W tym celu należy użyć opcji `[-d]`:

```
./bids -n 192.168.0.0 -m 255.255.255.0 -d net 192.168.0.0
```

Alerty o wykrytych atakach będą wypisywane na standardowe wyjście. Domyślnie, alerty o prawdopodobieństwie minimum 0.75 są drukowane na ekran. Wartość tą można zmienić przed kompilacją w pliku `config.h` (8.3.2.2). Przykładowy alert ma następującą postać:

```
Atak 'tcp_syn_flood' z prawdopodobienstwem 0.987 z 192.168.10.10:0 ->
192.168.0.1:70
```

W kolejnych polach system drukuje rodzaj wykrytego ataku, obliczone prawdopodobieństwo, adres systemu, z którego następuje atak oraz system docelowy. W przypadku protokołów UDP/TCP są dodatkowo podawane porty źródłowe/docelowe, a w przypadku protokołu ICMP – typ serwisu. System uogólnia informacje, dlatego w alertach może zamiast konkretnego adresu IP pojawić się adres składający się z samych zer (0.0.0.0):

```
Atak 'tcp_syn_flood' z prawdopodobieństwem 0.987 z 0.0.0.0:0 ->
192.168.0.1:70
```

Taki rekord oznacza, że całosciowy ruch wygenerowany na dany adres IP i port w chronionej sieci nosi znamiona ataku typu DoS/DDoS. Adres 0.0.0.0 oznacza po prostu dowolny adres. Podobnie, do analizy nie są brane źródłowe numery portów maszyn spoza sieci chronionej. W ich przypadku zawsze występuje wartość 0. Kolejną kwestią związaną z drukowaniem ostrzeżeń, jest sytuacja, kiedy to źródłem prawdopodobnego ataku DoS/DDoS jest maszyna w chronionej sieci:

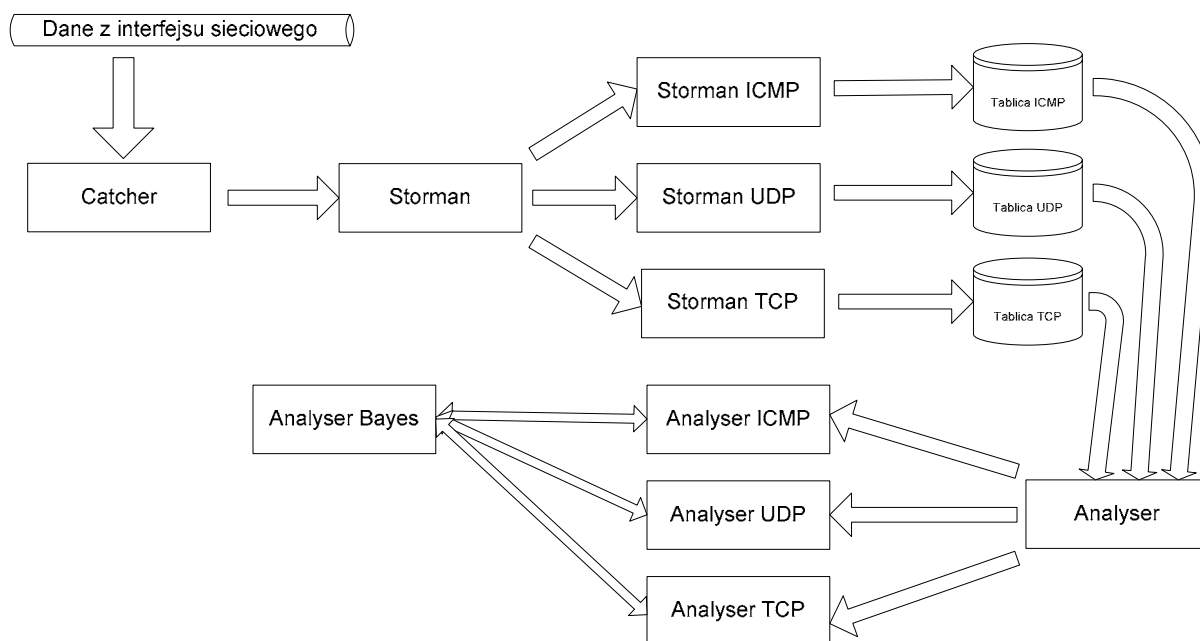
```
Atak 'udp_domain_flood' z prawdopodobieństwem 0.929 z 192.168.0.1:53 ->
192.168.10.10:0
```

Powyższa linia sugeruje, że ruch z naszej chronionej sieci przekracza normy i nosi znamiona ataku DoS/DDoS. Takie linie mogą wskazywać na atak na serwer DNS, lub pośrednictwo naszej sieci w ataku z wykorzystaniem powielaczy. (6.3)

### 8.3. Opis techniczny

#### 8.3.1. Moduły wewnętrzne

W celu funkcjonalnego podziału i lepszego zarządzania kodem, system BIDS składa się z kilkunastu modułów. Główne moduły wraz z kierunkiem przepływu danych zostały przedstawione na poniższym rysunku.



Rysunek 37 Moduły systemu BIDS i kierunki przepływu danych

W kolejnych podrozdziałach zostaną opisana funkcjonalność poszczególnych modułów.

#### 8.3.1.1. Moduł Catcher

Jest to główna część systemu. Jego zadaniami w kolejności wykonywana są:

- Pobranie, sprawdzenie poprawności oraz przetworzenie parametrów linii poleceń (8.2.3)
- Wyświetlenie krótkiej pomocy w przypadku braku wymaganego parametru
- Inicjalizacja nasłuchiwanie poprzez wywołanie biblioteki PCAP [49] (pcap\_init).
- Pobranie parametrów interfejsu, na którym następuje nasłuchiwanie (pcap\_dloff).
- Wywołanie funkcji inicjalizującej Storman'a (storman\_init).
- Wywołanie funkcji inicjalizującej Analyser'a (analyser\_init)
- Wejście do nieskończonej pętli (tcp\_rst\_loop)
  - Odebranie pakietu korzystając z funkcji biblioteki PCAP (pcap\_next)
  - Rozpoznanie typu protokołu (ICMP, UDP, TCP, inny)
  - Przepisanie danych pakietu do specjalnej struktury (12.2)
  - Przekazanie danych do modułu Storman.

Moduł wykorzystuje bibliotekę PCAP z pakietu TCPDUMP [49]. Jest najlepsza biblioteka w systemach klasy UNIX, która dostarcza łatwego interfejsu do pobierania pakietów z karty sieciowej w normalnym trybie, jak i w trybie testowym (PROMISC). Dlatego wykorzystywana jest przez większość analizatorów ruchu jak i systemów IDS (Ethereal, Tcpcdump, Snort), również na platformie WIN32. Listę funkcji tej biblioteki zamieszczono w dodatku (12.4).

#### 8.3.1.2. Moduł Storman

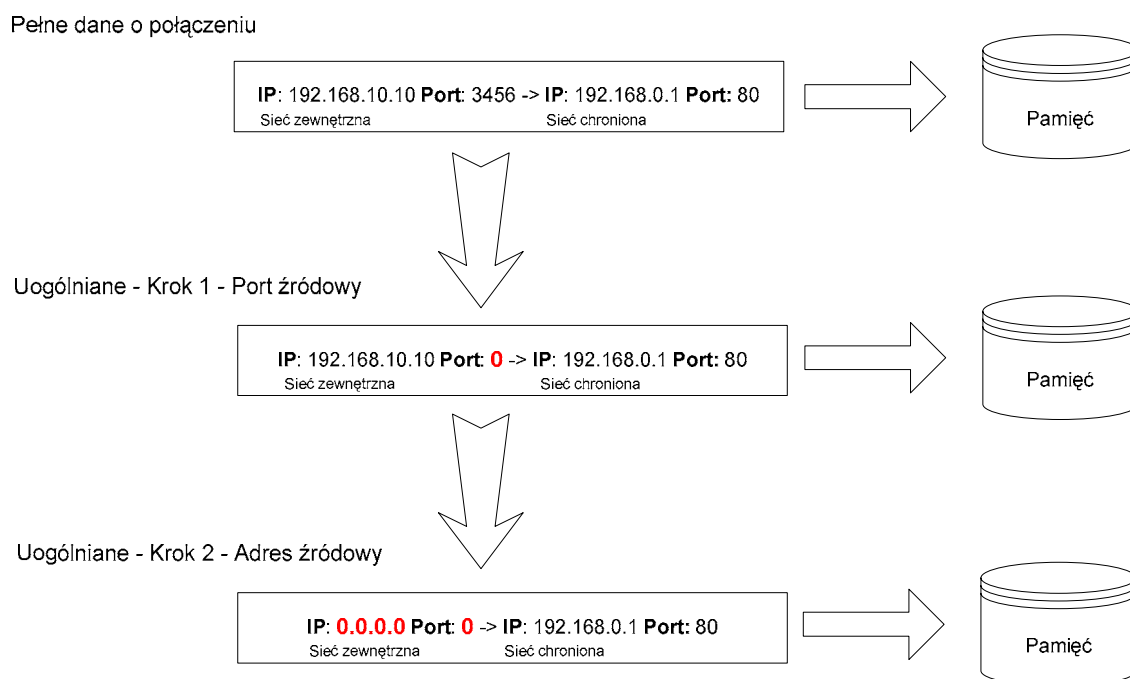
Zadaniem modułu Storman jest przechowywanie i zarządzanie informacjami o połączeniach. Jego cykl pracy jest podobny w przypadku wszystkich protokołów i przedstawia się następująco:

- Odebranie paczki danych od Catcher'a.
- Przeszukanie sekwencyjne tablicy segmentów w poszukiwaniu pasującego, istniejącego rekordu.
- Jeśli rekord zostanie znaleziony, dodatnie do niego rozmiaru aktualnej paczki danych i zwiększenie liczby pakietów o 1.

- Jeśli rekord nie został znaleziony, wyszukiwane jest pierwsze wolne miejsce i w nim paczka jest zapisywana lub alokowany jest nowy segment danych.
- Uogólnia się informacje o pakiecie i powtarza cykl od początku.

Moduł Storman wywołuje bliźniacze funkcje znajdujące się w modułach Storman – ICMP, Storman – UDP, Storman – TCP, które obsługują specyficzne pola nagłówka dla danego protokołu. Wszystkie dane są zapisywane w pamięci operacyjnej. Rozmiar pojedynczego rekordu nie przekracza 100 bajtów, dlatego nawet przy bardzo obciążonych sieci i dużej ilości połączeń całkowita zaalokowana pamięć nie przekraczała podczas testów 10 MB. Zaletą trzymania danych w pamięci jest ogromna szybkość przetwarzania.

Do dalszej analizy dane muszą być **uogólniane**. Postępowanie polega na odrzucaniu po kolei portu oraz adresu nadawcy (z poza sieci chronionej) i następnie zsumowaniu danych z innymi połączeniami. Pozwoli to na wykrywanie ataków rozproszonych DDoS, oraz ataków na usługę w sieci chronionej, gdzie często adres i port źródłowy są losowane.



Rysunek 38 Schemat postępowania przy uogólnianiu danych

Powyższy przykład dotyczy protokołu UDP i TCP. Dla protokołu ICMP schemat postępowania jest podobny.

### 8.3.1.3. Moduł Analyser

Moduł Analyser jest sercem systemu BIDS. Jego zadaniem jest analiza zgromadzonych danych przechowywanych w pamięci operacyjnej. Moduł działa w postaci wątku, osobno od



reszty modułów. Wątek został oprogramowany korzystając z biblioteki POSIX Threads [52]. Celem takiej implementacji było uniezależnienie go od reszty programu i dowolne dostosowanie tempa i czasu przetwarzania. Kolejność postępowania jest następująca:

- Wywołanie odpowiedniej funkcji z modułu specyficznego dla danego protokołu
  - Wyszukanie sekwencyjne ważnego rekordu, którego ostatnia analiza była dawniej niż zdefiniowany interwał `MIN_TIME_ANALYSE` (8.3.2.2).
  - Zablokowanie rekordu korzystając z `MUTEX`'ów zawartych w każdym rekordzie i funkcji `pthread_mutex_lock`. Blokada zabezpiecza przed jednoczesną modyfikacją danych przez Storman'a.
  - W zależności od trybu pracy: obliczenie wiersza pliku `CASE` (8.3.2.4), wyświetlenie wiersza pliku obserwacji sieci (8.3.2.3), czy wreszcie przekazanie danych do modułu `Analyser – Bayes` w celu obliczenia prawdopodobieństwa ataku.
  - Wyzerowanie danych dotyczących znalezionej analizy (suma rozmiarów, ilość) oraz odblokowanie rekordu.
  - Wyszukanie następnego rekordu do analizy.
- Po wywołaniu odpowiednich funkcji dla trzech protokołów, mierzony jest sumaryczny czas analizy. Jeśli ten czas jest mniejszy niż zadeklarowany `MIN_INTERVAL_ANALYSE` (8.3.2.2) wątek usypia na różnicę tych czasów. Takie działanie ma na celu odciążenie procesora maszyny, przez unikanie bezsensownych obiegów pętli.

#### 8.3.1.4. Moduł `Analyser – Bayes`

Moduł ten jest mózgiem całego systemu. Odpowiada on za załadowanie zgromadzonych danych do sieci Bayesa i odczytanie prawdopodobieństwa.

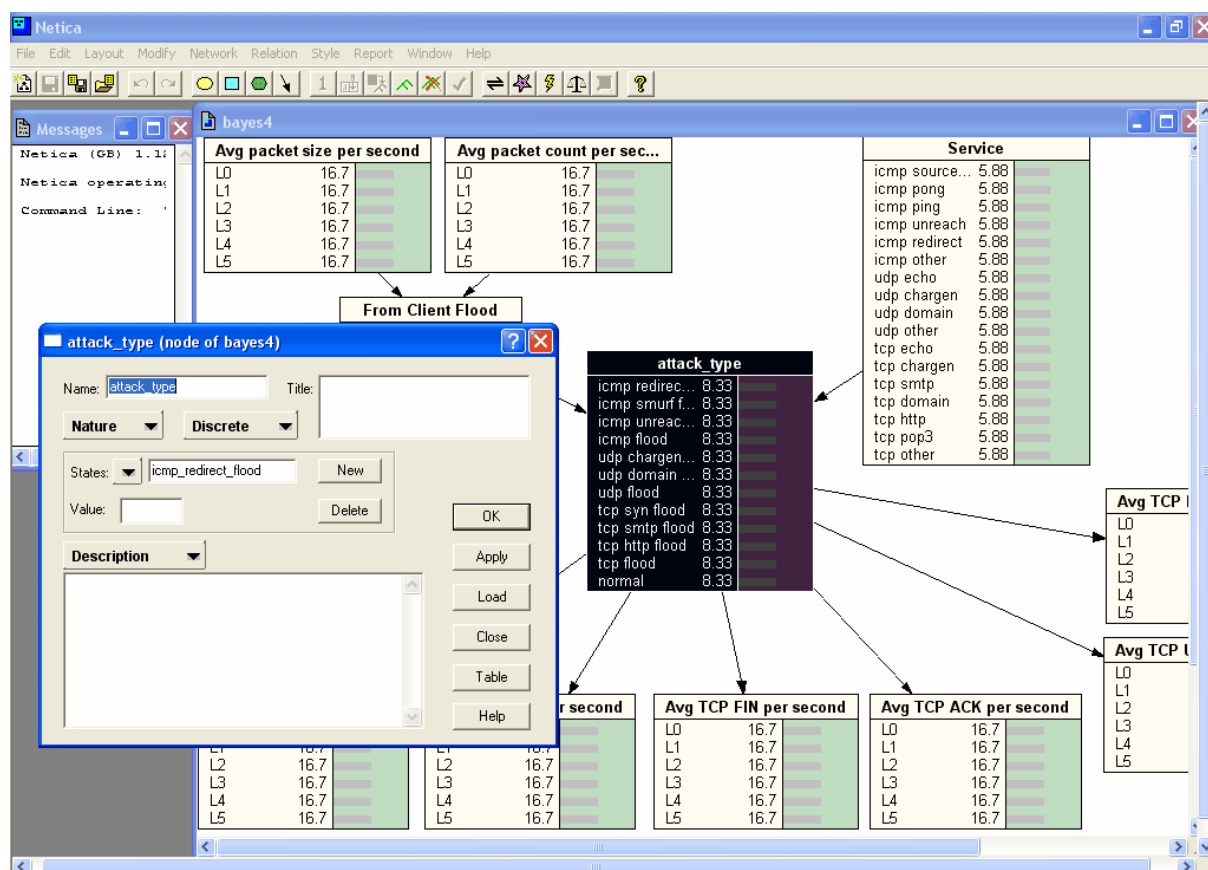
U podstaw koncepcja sieci Bayesa leży twierdzenie podane przez angielskiego matematyka Thomas Bayes (1702–61). W rzeczywistym świecie jest wiele sytuacji, w których wystąpienie jakiegoś zdarzenia ściśle zależy od innego zdarzenia. Zastosowanie sieci Bayesa pozwala na uniknięcie obliczeń o dużej złożoności – obliczenie jednego prawdopodobieństwa a posteriori łączy się z uprzednim obliczeniem wykorzystywanych prawdopodobieństw. Ogromną zaletą sieci Bayesa jest przedstawianie wiedzy niepewnej, zdarzeń, co, do których nie ma pewności, że zaszły. Klasyczna sieć Bayesa składa się z węzłów, które reprezentują zmienne oraz łuków definiujących powiązania pomiędzy węzłami. Graficznie przedstawienie sieci przybiera postać acyklicznego grafu.

W dziedzinie wykrywania ataków sieć Bayesa posiada kilka ważnych zalet:

- Bardzo duża szybkość obliczeń – w przypadku pracy systemu BIDS na bardzo obciążonej sieci, może być potrzeba setek analiz na sekundę. Obliczanie prawdopodobieństw wewnątrz sieci Bayesa jest proste i szybkie. System nie może pozwolić sobie na zajęcie czasu procesora na poziomie 100%, gdyż sam stałby się narzędziem DOS.
- Możliwość uczenia się – sieci Bayesa pozwalają na szybkie uczenie się. Jedynie, co trzeba im dostarczyć to stany zmiennych w rzeczywistych zdarzeniach.
- Przetwarzanie wiedzy niepewnej – w przypadku pewnych ataków, lub ruchu prawidłowego stan niektórych węzłów pozostaje nieznan.

Jako silnik implementujący sieć Bayesa została wybrany pakiet Netica firmy Norsys [51]. Przyczynami takiego wyboru były:

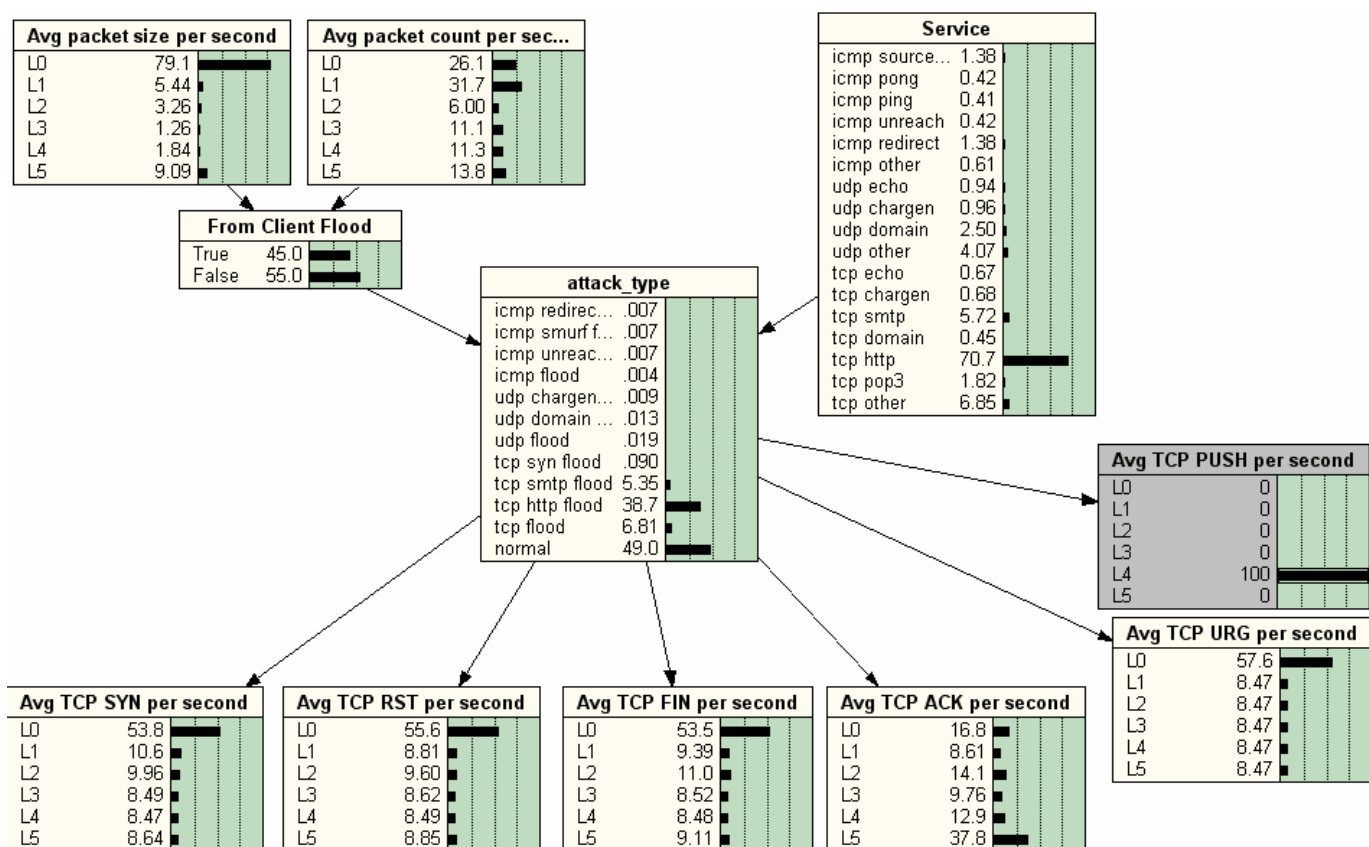
- Wersja graficzna pakietu pod system Windows
- Biblioteki C oraz Java zarówno na platformę Linux jak i Windows
- Dobrze zaprojektowane API
- Stabilność pakietu
- Darmowa wersja limitowana (limitowanie dotyczy ilości węzłów)



Rysunek 39 Graficzny moduł pakietu Netica (Windows)

Projekt sieci Bayesa został wykonany po przeanalizowaniu kilkunastu innych sieci z uwzględnieniem zależności pomiędzy gromadzonymi wartościami. Stany wszystkich węzłów mają charakter dyskretny. Ze względu na gromadzone dane, zostały zdefiniowane następujące węzły w sieci Bayesa:

- apsp (Avg packet size per second) - średnia ilość przesłanych danych w skali od L0...L5 (na sekundę).
- apcp (Avg packet count per second) – średnia ilość przesłanych pakietów w skali od L0...L5 (na sekundę).
- fcf (From Client Flood) – węzeł wskazujący, czy następuje powódź pakietów.
- srv (Service) – rodzaj usługi w sieci chronionej do jakiej kierowane są pakiety. Wybrano popularne usługi, które działają na protokołach: ICMP, UDP, TCP. Reszta usług będzie pokrywana przez stany węzła: icmp\_other, udp\_other i tcp\_other.
- atsp (Avg TCP SYN) – średnia ilość pakietów protokołu TCP mających zapaloną flagę SYN w stali L0...L5 (dla ICMP i UDP zawsze L0).
- atrp (Avg TCP RST) – średnia ilość pakietów protokołu TCP mających zapaloną flagę RST w stali L0...L5 (dla ICMP i UDP zawsze L0).
- atfp (Avg TCP FIN) – średnia ilość pakietów protokołu TCP mających zapaloną flagę FIN w stali L0...L5 (dla ICMP i UDP zawsze L0).
- atap (Avg TCP ACK) – średnia ilość pakietów protokołu TCP mających zapaloną flagę ACK w stali L0...L5 (dla ICMP i UDP zawsze L0).
- atpp (Avg TCP PUSH) – średnia ilość pakietów protokołu TCP mających zapaloną flagę PUSH w stali L0...L5 (dla ICMP i UDP zawsze L0).
- atup (Avg TCP URG per second) – średnia ilość pakietów protokołu TCP mających zapaloną flagę URG w stali L0...L5 (dla ICMP i UDP zawsze L0).
- attack\_type – węzeł, który jako swoje stany posiada nazwy ataków. Ostatni stan o nazwie 'normal', oznacza brak ataku.



Rysunek 40 Schemat sieci Bayesa

#### 8.3.1.4.1. Funkcja dyskretyzacji

Dane o ilości pakietów oraz ich rozmiarze, które są przechowywane w pamięci mają charakter ciągły – są to liczby rzeczywiste. Kluczowym fragmentem analizy jest ich zamiana na specjalną skalę 6-stopniową L0...L5, która jest wymagana przez sieć Bayesa. Do tego celu są wykorzystywane 3 funkcje dyskretyzujące, każda w innej sytuacji.

- Pierwsza z nich stosowana jest do zamiany ilości pakietów. Jest stosowana do wszystkich protokołów. Funkcja ma charakter logarytmiczny, rośnie szybko na początku, łagodniej dla większych argumentów i dlatego została wybrana do przeliczania ilości pakietów.

$$L = \text{round}(\log_{nc+1} \frac{1}{\text{MID\_LEVEL}} (count + 1))$$

IF L > MAX\_LEVEL THEN L = MAX\_LEVEL

gdzie,

L – obliczony poziom MIN\_LEVEL...MAX\_LEVEL

nc – współczynnik odczytany z pliku net\_stat.conf (\_FROM\_NETWORK\_COUNT,

\_TO\_NETWORK\_COUNT), inny dla każdego w protokołów. (8.3.2.1)

MID\_LEVEL – średni poziom zdefiniowany w pliku config.h. (8.3.2.2)

count – ilość pakietów w tym rekordzie danych (przechwyconych)

MAX\_LEVEL – maksymalny poziom zdefiniowany w pliku config.h. (8.3.2.2)

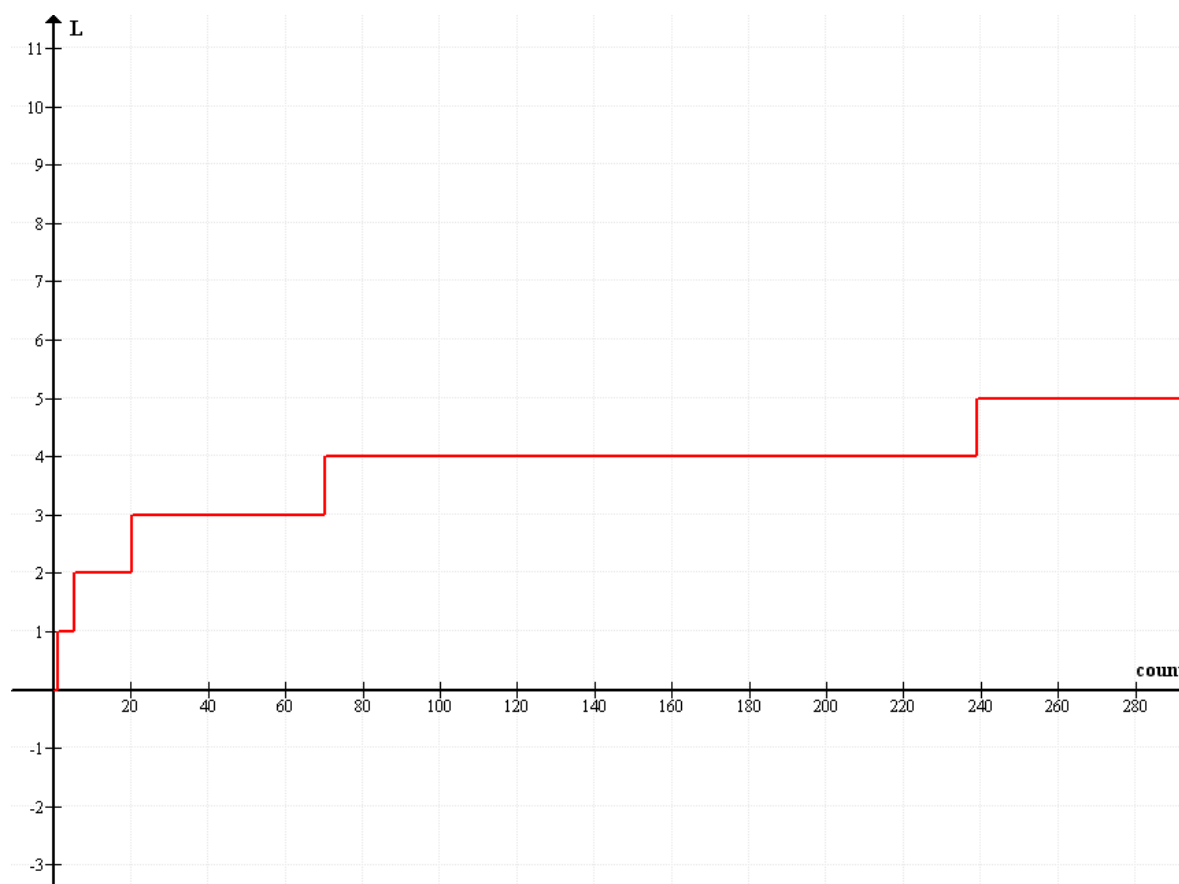
Dla:

nc=20

MID\_LEVEL=2.5

MAX\_LEVEL=5

Wykres funkcji jest następujący:



Rysunek 41 Wykres funkcji dyskretyzującej 1

- Druga z nich, również jest stosowana do wszystkich protokołów i przelicza rozmiar zgromadzonych danych na poziom L0...L5. Ma charakter liniowy.

$$L = \text{round}\left(\frac{\text{size} * \text{MID\_LEVEL}}{ps}\right)$$

IF L > MAX\_LEVEL THEN L = MAX\_LEVEL

gdzie,

$L$  – obliczony poziom  $\text{MIN\_LEVEL} \dots \text{MAX\_LEVEL}$

size – rozmiar przechwyconych pakietów

ps – współczynnik odczytany z pliku `net_stat.conf` (`_FROM_NETWORK_SIZE`, `_TO_NETWORK_SIZE`), inny dla każdego w protokołów. (8.3.2.1)

$\text{MID\_LEVEL}$  – średni poziom zdefiniowany w pliku `config.h`. (8.3.2.2)

$\text{MAX\_LEVEL}$  – maksymalny poziom zdefiniowany w pliku `config.h`. (8.3.2.2)

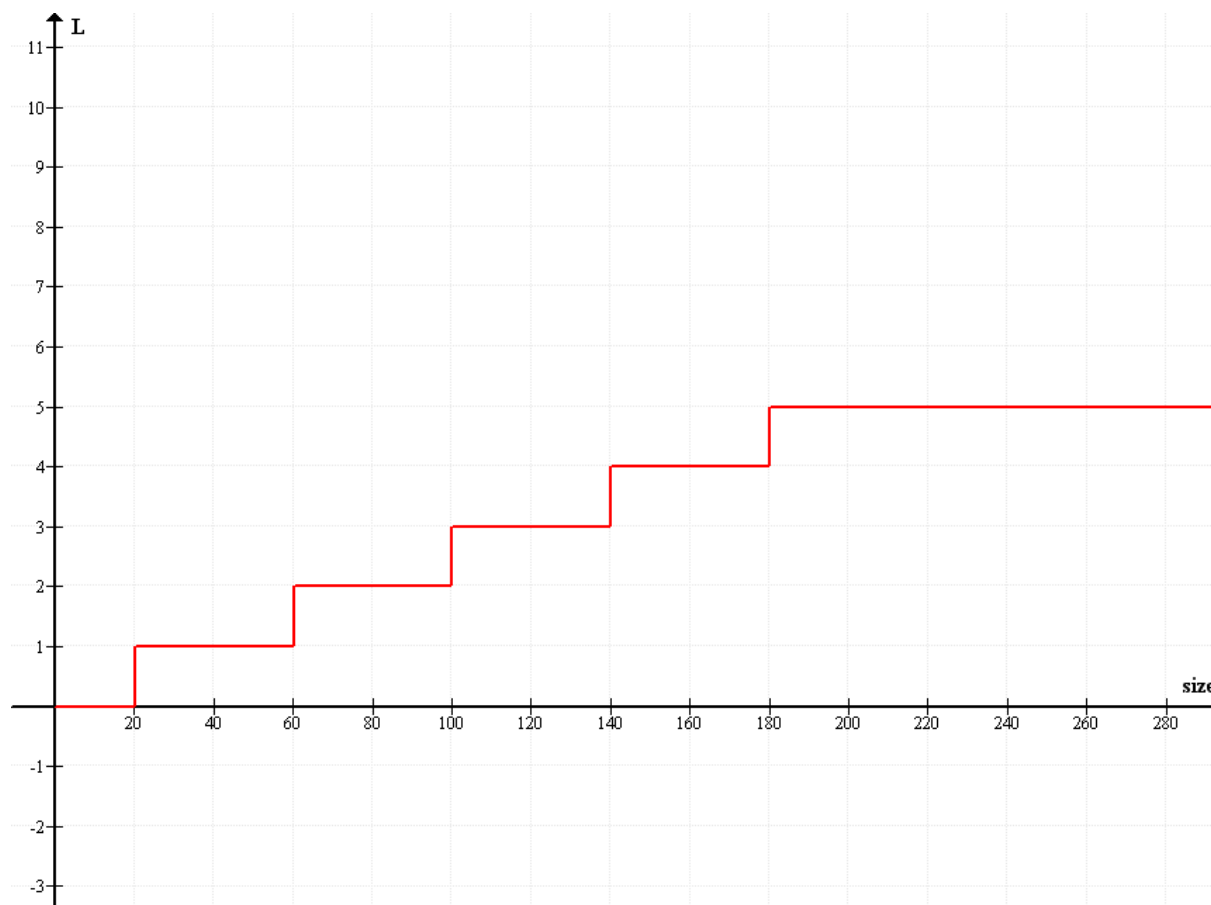
Dla:

ps=100

$\text{MID\_LEVEL}=2.5$

$\text{MAX\_LEVEL}=5$

Wykres funkcji jest następujący:



Rysunek 42 Wykres funkcji dyskretyzującej 2

- Ostatnią funkcją ma za zadanie przeliczyć występowanie flag w pakietach TCP (SYN, ACK itp), na odpowiednie poziomy.

$$L = \text{round}\left(\frac{\text{flag} * \text{MAX\_LEVEL}}{\text{count}}\right)$$

gdzie,

L – obliczony poziom MIN\_LEVEL...MAX\_LEVEL

MAX\_LEVEL – maksymalny poziom zdefiniowany w pliku config.h. (8.3.2.2)

flag – ilość pakietów z zapaloną daną flagą (przechwyconych)

count – ilość pakietów w tym rekordzie danych (przechwyconych)

#### 8.3.1.4.2. Uczenie sieci Bayesa

Sieć Bayesa składa się z węzłów, które reprezentują zmienne, oraz łuków – prawdopodobnych zależności pomiędzy węzłami. Sieć charakteryzuje się tym, że poprzednik węzła bezpośrednio powoduje stan zmiennej skojarzonej z tym węzłem. Dla każdego węzła określa się rozkład prawdopodobieństwa warunkowego. Dla wartości dyskretnych funkcję prawdopodobieństwa warunkowego można zapisać w tabeli przedstawiającej prawdopodobieństwa przyjęcia poszczególnych wartości (własnych) przez węzeł dziecko, w zależności od kolejnych z możliwych wartości rodzica.

apsps	apcps	True	False
L0	L0	0.0286	99.971
L0	L1	0.0434	99.957
L0	L2	5.248	94.752
L0	L3	99.797	0.203
L0	L4	99.180	0.820
L0	L5	99.569	0.431
L1	L0	0.0861	99.914
L1	L1	0.0101	99.990
L1	L2	2.752	97.248
L1	L3	75.929	24.071
L1	L4	99.870	0.130
L1	L5	99.716	0.284
L2	L0	4.167	95.833
L2	L1	14.269	85.731
L2	L2	1.343	98.657

Rysunek 43 Przykładowy rozkład prawdopodobieństw w węźle sieci Bayesa

Podczas tworzenia systemu BIDS przyjęto, że sieć zostanie nauczona rozpoznawać ataki typu DoS/DDoS i prawdopodobieństwa nie będą ręcznie poprawiane. Przygotowanie odpowiednich pików uczących pozwoliło zrealizować to założenie.

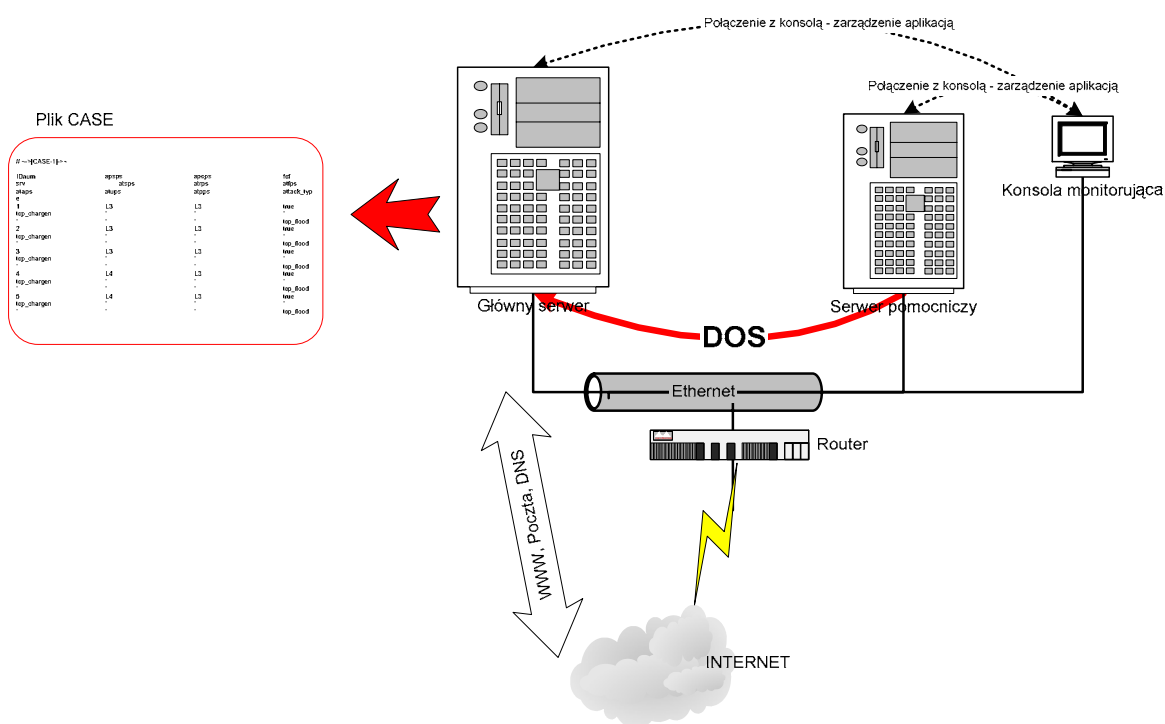
- Serwer WWW
- Serwer poczty (SMTP, POP3, IMAP)
- Serwer DNS

- W pierwszej fazie system BIDS został uruchomiony z przełącznikiem [-c], czyli z opcją generowania pliku CASE. System w tym trybie obserwuje cały wychodzący i wchodzący ruch, agreguje dane i wypisuje wyniki na ekran. Wypisywanie wiersze mają tzw. stan normalny, co oznacza, że zostały na sztywno zakwalifikowane jako ruch prawidłowy (nie atak). Jednocześnie cały czas ruch wychodzący i wchodzący do serwera był obserwowany innym narzędziem (iptraf, MRTG), aby sprawdzić czy nie następuje próba ataku. Jeśli nastąpiłby prawdziwy atak, został by on przez system BIDS w tym trybie zakwalifikowany jako prawidłowy ruch. W rezultacie sieć dostała by błędne dane. Rezultatem tej fazy był plik CASE o rozmiarze 47MB i zawierający 319748 rekordów. Część tego pliku jest w dodatku (12.2).



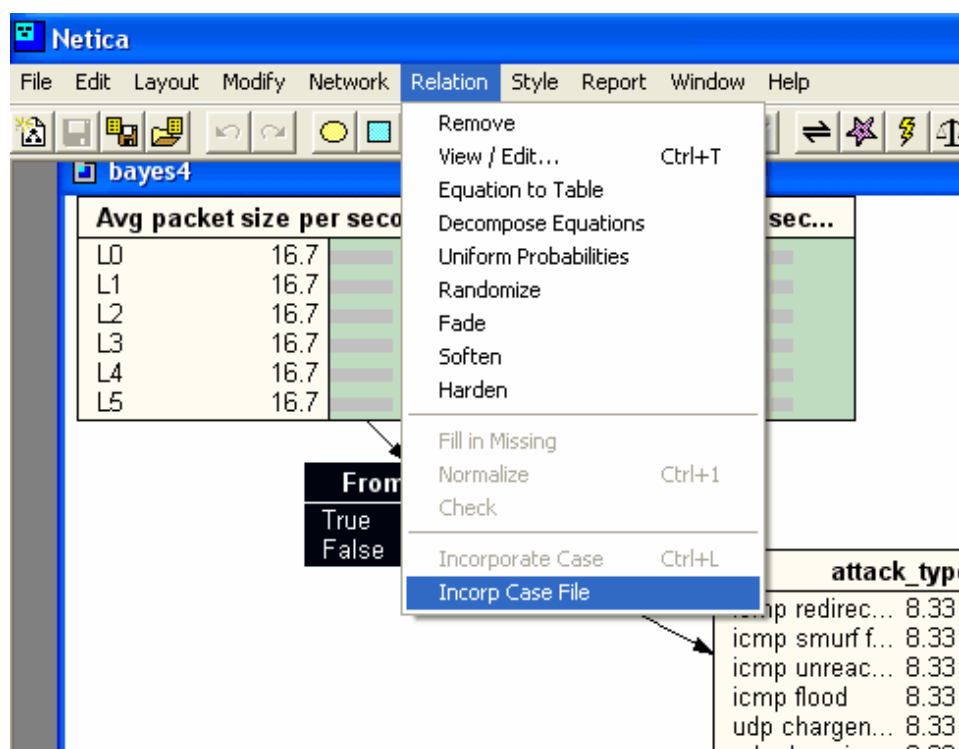


- Druga faza miała na celu nauczenie sieci Bayesa wykrywania ataków DoS/DDoS. W tym celu na tym samym serwerze system BIDS został uruchomiony z przełącznikiem [-a]. Przełącznik ten modyfikuje wyjściowy plik CASE w taki sposób, że kolejne dane są oznaczone jako ataki. Dodatkowo dodano odpowiedni filtr do systemu BIDS, który przepuszczał ruch tylko z jednej podsieci. Do nauki został użyty drugi serwer znajdujący się w sieci lokalnej. Z niego, przy wykorzystywaniu specjalnych narzędzi były symulowane ataki DoS oraz DDoS. Do części ataków stosowany był program hping2 [54], do innych zostały napisane specjalne skrypty w języku PERL i biblioteki Net::RawIP. Symulowane były kolejne ataki z różną siłą – zmianie ulegało natężenie pakietów oraz ich rozmiar. Najlepiej, aby nauka nie odbywała się w sieci lokalnej, jednak nie było możliwości jej przeprowadzenia z komputera zdalnego, gdyż zostaliby odcięci klienci obu sieci. Rezultatem tej fazy były 4 pliki CASE: dla protokołu ICMP, UDP, TCP oraz osobny plik z wykrytymi atakami typu SYN Flood.



Rysunek 45 Schemat sieci podczas drugiej fazy uczenia

Wszystkie pliki z obu faz zostały następnie wczytane do pakietu NETICA, który dokonał ich przetworzenia.



Rysunek 46 Opcja uczenia pakietu Netica

Podczas importu pakiet umożliwia wybranie tzw. stopnia importu. Jest to liczba całkowita, większa od 0. Wartość ta determinuje, ale razy każdy wiersz ma być przetwarzany. Jeśli plik CASE ma 10 przypadków, a podany współczynnik jest równy 5, wtedy każda linia jest 5 razy wpływa na zmianę prawdopodobieństw. Pliki CASE z atakami zawierały o wiele mniej danych niż plik CASE z prawidłowym ruchem, dlatego każdy z nich został zaimportowany z współczynnikiem 5. Więcej informacje o algorytmach uczenia sieci Bayesa można znaleźć w pomocy pakietu Netica [51] oraz w publikacjach [53] [55] [56].

### 8.3.2. Formaty plików

#### 8.3.2.1. Plik net\_stat.conf

Plik ten przechowuje informację o obliczonej statystyce sieci – średniej ilości wysłanych i odebranych pakietów. Tworzony jest on przez skrypt net\_stat\_update (8.2.4). Format pliku jest ściśle określony i ma następujący format:

```
ICMP_TO_NETWORK_COUNT=1.000000
ICMP_FROM_NETWORK_COUNT=1.000000
ICMP_TO_NETWORK_SIZE=1.000000
ICMP_FROM_NETWORK_SIZE=1.000000
```

```
UDP_TO_NETWORK_COUNT=1.000000
UDP_FROM_NETWORK_COUNT=1.000000
UDP_TO_NETWORK_SIZE=1.000000
UDP_FROM_NETWORK_SIZE=1.000000
```

```
TCP_TO_NETWORK_COUNT=1.000000
TCP_FROM_NETWORK_COUNT=1.000000
TCP_TO_NETWORK_SIZE=1.000000
TCP_FROM_NETWORK_SIZE=1.000000
```

Dla każdego z protokołów (ICMP, UDP, TCP) są cztery wiersze.

- `_TO_NETWORK_COUNT` – średnia ilość pakietów przesłana do chronionej sieci na sekundę.
- `_FROM_NETWORK_COUNT` – średnia ilość pakietów przesłana z chronionej sieci na sekundę.
- `_TO_NETWORK_SIZE` – średnia wielkość pakietów w KB przesłana do chronionej sieci na sekundę.
- `_FROM_NETWORK_SIZE` – średnia wielkość pakietów w KB przesłana z chronionej sieci na sekundę.

Wielkości są liczbami rzeczywistymi z 6 miejscami po przecinku.

### 8.3.2.2. Plik config.h

Plik `config.h` zawiera definicje głównych parametrów, które są ustawiane podczas kompilacji. Część z nich można zmienić dostosowując do własnych potrzeb, natomiast przypadkowa zmiana innych spowoduje błędne działanie aplikacji. Dlatego definicje zostały podzielone na dwie grupy: podstawowe i zaawansowane.

Tabela 5 Podstawowe parametry w pliku config.h

Nazwa	Opis	Wartość domyślna
<code>SLOG_LEVEL</code>	Poziom logowania; liczba w przedziale od 0-5. Wyższa wartość oznacza bardziej szczegółowe informacje.	0
<code>ARROFARR_MAX</code>	Ilość segmentów danych. W 1000 segmentów można zapamiętać 1000*1000	1000

	połączeń.	
ARRAY_SIZE	Rozmiar pojedynczego segmentu danych. Liczba oznacza ilość rekordów w tym segmencie. Jeden rekord służy do zapamiętania jednego połączenia.	1000
STOR_EXPIRE	Czas w sekundach, po których dany rekord z opisem połączenia przestaje być ważny i może być nadpisany. Czas liczy się od ostatniego przyjscia pakietu z tego połączenia.	60
MIN_TIME_ANALYSE	Jest to minimalny czas, jaki musi nastąpić pomiędzy analizą tego samego rekordu. Zapobiega to przed zbyt częstą analizą tego samego rekordu.	3
MIN_INTERVAL_ANALYSE	Jest to minimalny czas, jaki musi upłynąć pomiędzy cyklem analizy. Jeśli analiza trwała poniżej tego czasu, aplikacja usypia na czas różnicy.	3
BAYESIAN_MIN_PROB	Minimalne prawdopodobieństwo wyliczone przez sieć Bayesa, które powoduje wydrukowanie ostrzeżenia.	0.75

Parametry zaawansowane:

Tabela 6 Zaawansowane parametry w pliku config.h

Nazwa	Opis	Wartość domyślna
MIN_LEVEL	Minimalny poziom w sieci Bayesa parametrów dyskretyzowanych.	0

MID_LEVEL	Pośrednia wartość poziomu w sieci Bayesa parametrów dyskretyzowanych.	2.5
MAX_LEVEL	Maksymalna wartość poziomu w sieci Bayesa parametrów dyskretyzowanych.	
slog_BUFF_SIZE	Rozmiar w bajtach buforu do wyświetlania komunikatów.	1024
VERSION	Wersja aplikacji	"1.0"
CONFIG_FILE	Nazwa pliku z zapisem statystyki sieci. (8.3.2.1)	"net_stat.conf"
BAYESIAN_FILE	Nazwa pliku z siecią Bayesa	"bids.dne"

### 8.3.2.3. Plik net.stat

Plik net.stat jest tworzony przez aplikację po zastosowaniu przełącznika [-t]. Ma on następujący format:

Protokół:Czas:Kierunek:Rozmiar:Ilo

Tabela 7 Format pliku net.stat

Protokół	Jedna z trzech wartości: ICMP, UDP, TCP
Czas	Bezwzględny czas w sekundach od daty 01.01.1970 z dokładnością do 6 miejsc po przecinku
Kierunek	Możliwe są dwie wartości: 0 – ruch do sieci chronionej, 1 – ruch z sieci chronionej
Rozmiar	Wielkość przesłanych danych w KB
Ilo	Ilość przesłanych pakietów

Przykładowe dane zostały umieszczone w załączniku (12)

#### 8.3.2.4. Plik CASE

Plik CASE jest zbiorem danych pozwalających na uczenie sieci Bayesa. Jego format jest ściśle określony przez firmę Norsys: Musi to być plik tekstowy, zawierający w jednej z 3 pierwszych linii ciąg znaków: „// ~->[CASE-1]->~”. Następnie muszą wystąpić wiersze zawierające etykiety kolejnych węzłów sieci Bayesa, oddzielone tabulatorami lub spacjami. Dodatkowo pierwszą kolumna musi być „IDnum” – rosnący zawsze o 1 licznik kolejnych wierszy. Przykładowy plik CASE może wyglądać następująco:

```
// ~->[CASE-1]->~
```

IDnum	apsps	apcps	fcf	srv	atsps	atrps	atfps	ataps	atups	atpps	attack_type
1	L3	L3	true	tcp_chargen	L0	*	L1	L2	L0	L1	tcp_flood
2	L3	L3	true	tcp_chargen	L0	L3	L1	L2	L0	L2	tcp_flood
3	L3	L3	true	tcp_chargen	L1	L3	L1	L2	L0	L3	tcp_flood
4	L4	L3	true	tcp_chargen	L2	L3	L1	*	L0	L5	tcp_flood

Specjalne znaczenie ma znak ‘\*’, który oznacza, że stan danego węzła nie jest znany. Fragment pliku CASE, który został użyty to nauki sieci Bayesa, można znaleźć w załączniku (12.2).

### 8.4. Analiza działania systemu BIDS

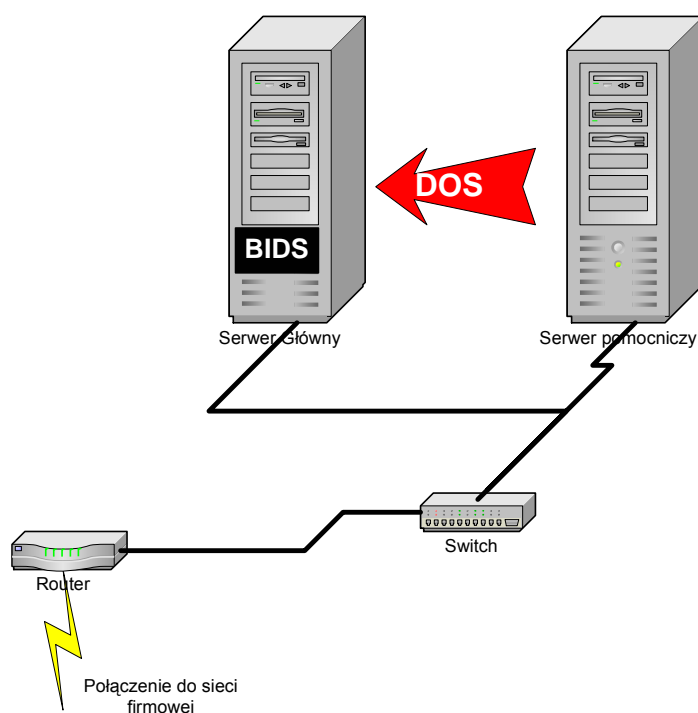
Celem tego rozdziału jest zmierzenie na ile wykonana aplikacja spełnia swoje zadanie. Testowane będą dwa ważne aspekty decydujące o przydatności systemu: skuteczność wykrywania ataków oraz wykorzystanie mocy obliczeniowej maszyny. Zmierzenie innych parametrów, typu ilość fałszywych alarmów nie była możliwa w warunkach laboratoryjnych (symulowanych ataków). Aby wyniki były wiarygodne, konieczna jest instalacja systemu w kilku produkcyjnych sieciach i testy trwające, co najmniej kilka miesięcy.

#### 8.4.1. Środowisko testowe

Środowiskiem testowym stała się sieć lokalna jednej z firm działających w branży IT. Sieć ta była odseparowana przy pomocy bramy (gateway) od produkcyjnych systemów tej firmy. Brak zakłóceń w funkcjonowaniu tychże systemów był głównym warunkiem udostępnienia infrastruktury. Środowisko testowe składało się z następujących elementów:

- Główny serwer testowy, na którym została zainstalowana aplikacja BIDS. Parametry komputera: Pentium II 400 MHz, 128 MB RAM, Dysk IDE 10 GB.

- Pomocniczy serwer testowy, który będzie symulował ataki DoS/DDoS na serwer główny. Parametry komputera: AMD XP 1400, 256 MB RAM, Dysk 40 GB.
- Przełącznik sieciowy 10/100 MBit 3COM
- Router Cisco, jego zadaniem było uniemożliwienie przedostania się generowanego ruchu do głównej sieci firmowej.
- System BIDS uruchomiony z przełącznikiem [-d], logowanie na poziomie 1 i 2 do pliku



Rysunek 47 Schemat środowiska testowego

Do generowania ruchu o pożądanym natężeniu i wielkości pakietów zostało wykorzystane narzędzie hping2 [54] oraz własne skrypty napisane w języku PERL i biblioteki Net::RawIP. Plik statystyki sieci zawierał następujące dane:

```
ICMP_TO_NETWORK_COUNT=11.765434
ICMP_FROM_NETWORK_COUNT=13.567754
ICMP_TO_NETWORK_SIZE=1.645444
ICMP_FROM_NETWORK_SIZE=1.567565

UDP_TO_NETWORK_COUNT=12.567855
UDP_FROM_NETWORK_COUNT=13.456234
UDP_TO_NETWORK_SIZE=1.353675
UDP_FROM_NETWORK_SIZE=5.347798
```

TCP\_TO\_NETWORK\_COUNT=54.345342

TCP\_FROM\_NETWORK\_COUNT=86.768767

TCP\_TO\_NETWORK\_SIZE=21.454456

TCP\_FROM\_NETWORK\_SIZE=54.376517

### 8.4.2. Testy skuteczności

Celem tych testów było zmierzenie skuteczności systemu, czyli jego zdolności do wykrywania ataków typu DoS/DDoS. W tym celu zasymulowano 3 ataków, po jednym z każdego protokołu (ICMP, UDP, TCP). Siła ataku będzie regulowana przez zwiększanie dwóch parametrów:

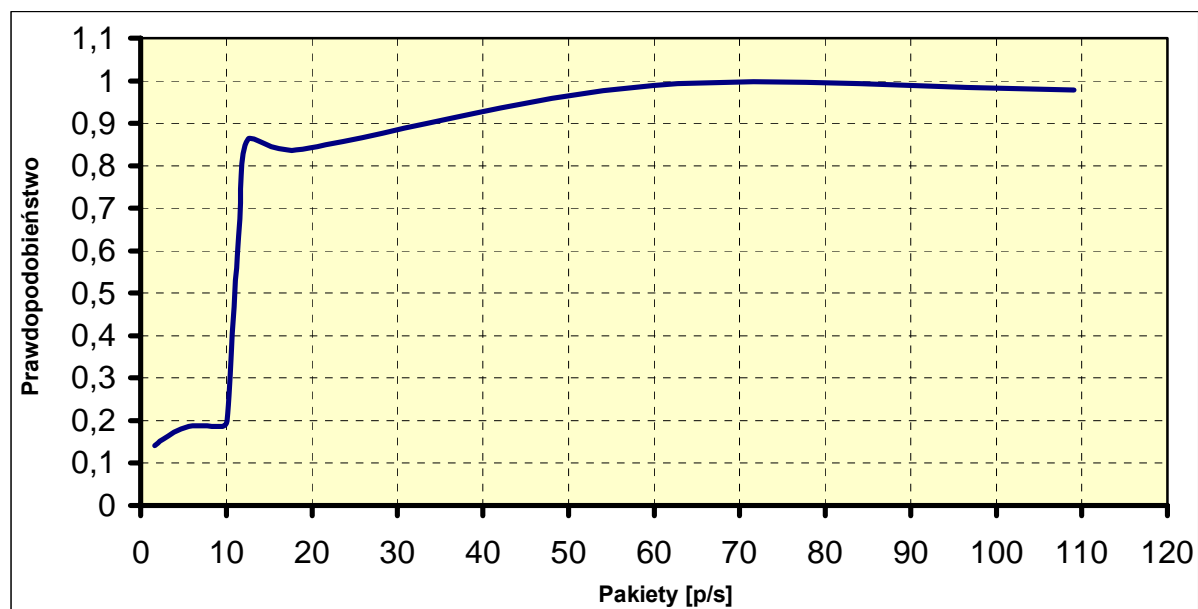
- Natężenia ilości pakietów [ilość/s]
- Natężenia ilości danych [KB/s]

Tabela 8 Zależność natężenia pakietów i prawdopodobieństwa ataku (ICMP Redirect)

Natężenie pakietów [p/s]	Prawdopodobieństwo
1,66	0,14
2,60	0,15
5,60	0,18
9,60	0,18
10,08	0,20
11,41	0,63
12,55	0,86
18,16	0,83
59,90	0,98



109,11	0,97
--------	------

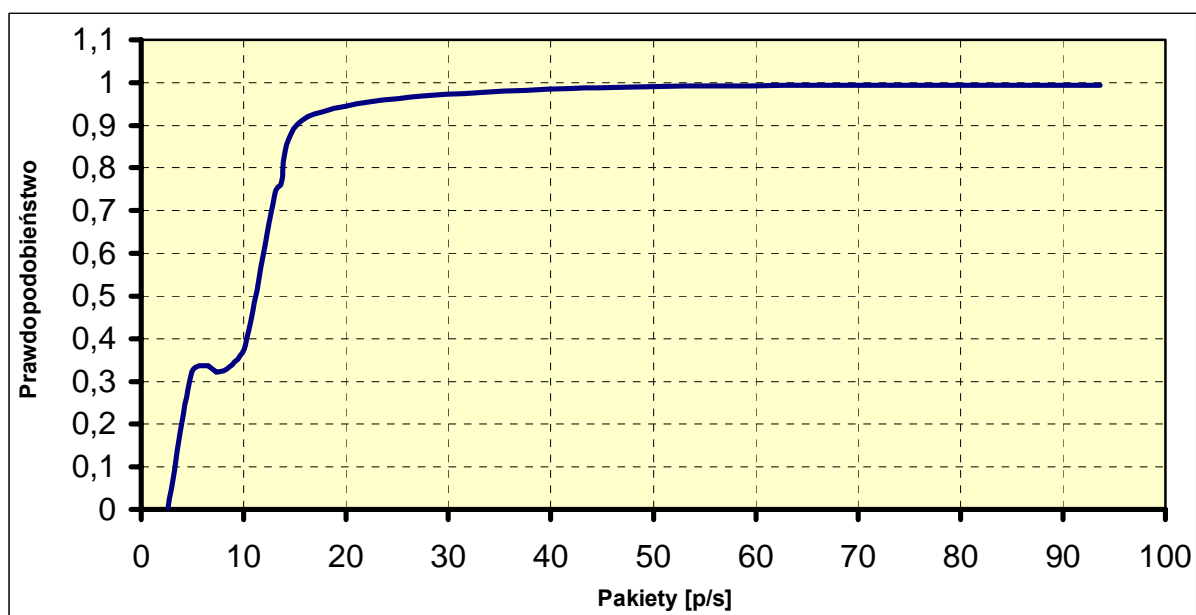


Rysunek 48 Test skuteczności – ICMP Redirect – rosnąca liczba pakietów

Tabela 9 Zależność natężenia pakietów i prawdopodobieństwa ataku (UDP Domain)

Natężenie pakietów [p/s]	Prawdopodobieństwo
2,60	0,00
4,95	0,32
6,40	0,33
7,45	0,32
8,74	0,33
10,20	0,38
13,14	0,74

13,62	0,76
16,87	0,92
42,02	0,98
93,65	0,99

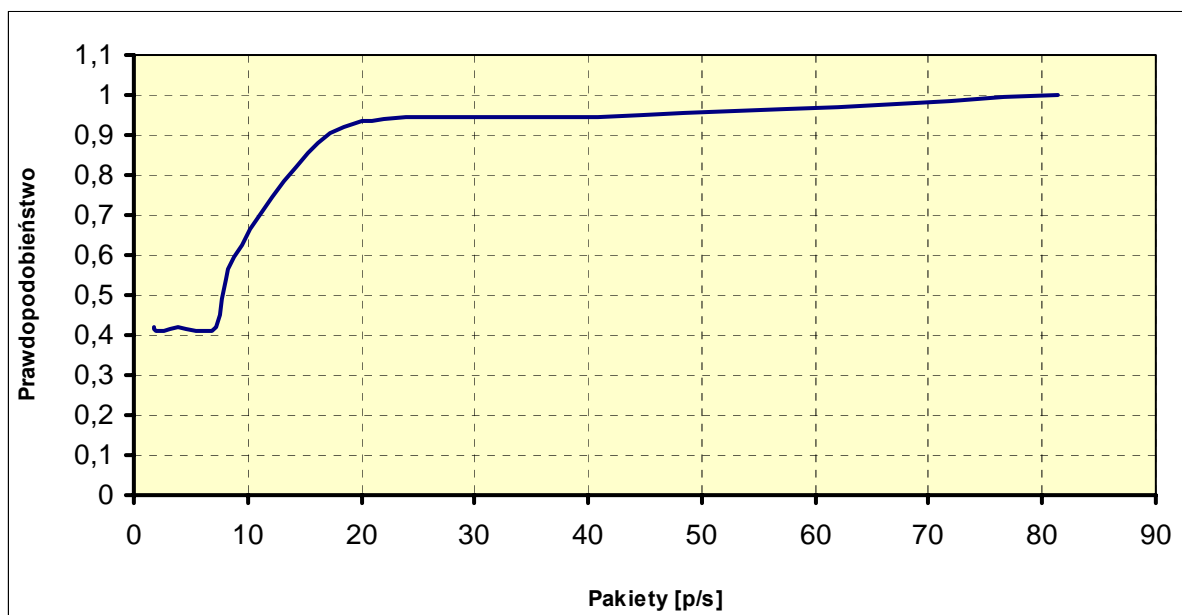


Rysunek 49 Test skuteczności – UDP Domain – rosnąca liczba pakietów

Tabela 10 Zależność natężenia pakietów i prawdopodobieństwa ataku (TCP SYN)

Natężenie pakietów [p/s]	Prawdopodobieństwo
1,78	0,41
1,88	0,40
2,69	0,41
3,80	0,41

7,20	0,41
8,84	0,59
16,12	0,88
22,00	0,93
44,62	0,95
81,33	0,99

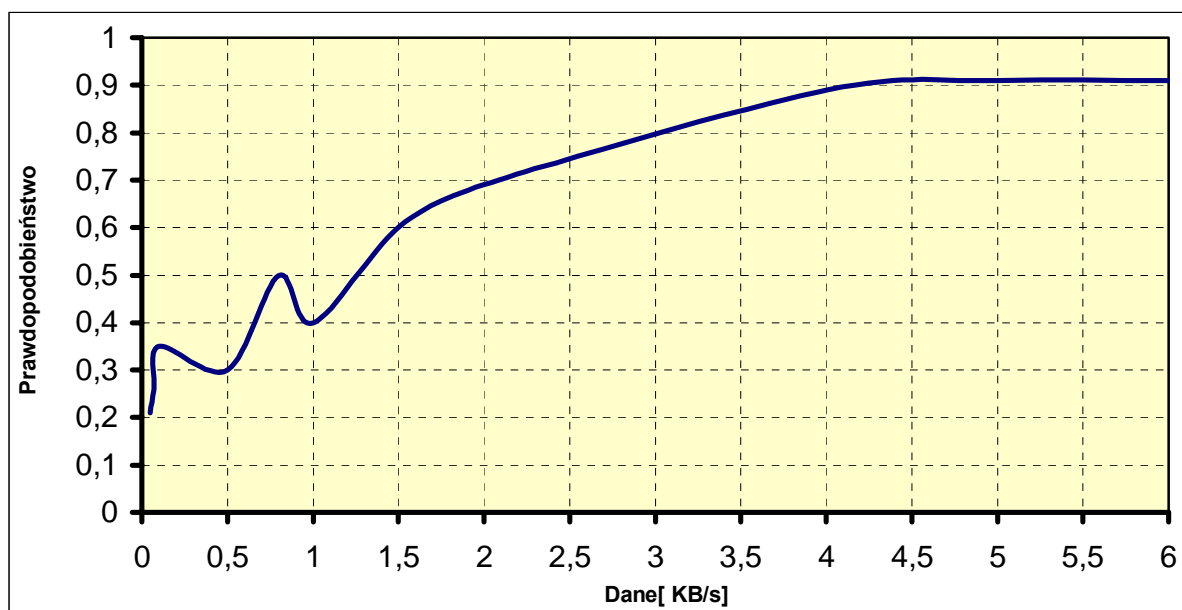


Rysunek 50 Test skuteczności – TCP SYN – rosnąca liczba pakietów

Tabela 11 Zależność natężenia danych i prawdopodobieństwa ataku (ICMP Redirect)

Natężenie danych [KB/s]	Prawdopodobieństwo
0,05	0,21
0,07	0,26
0,1	0,35

0,5	0,3
0,8	0,5
1	0,4
1,5	0,6
2	0,69
4	0,89
5	0,91
6	0,91

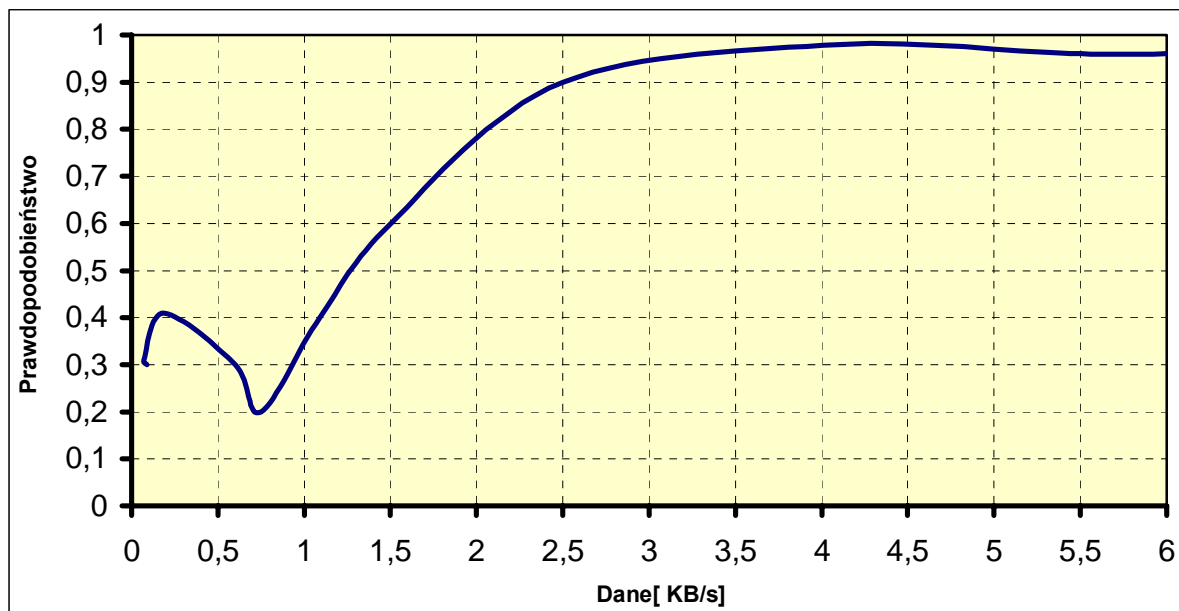


Rysunek 51 Test skuteczności – ICMP Redirect – rosnąca wielkość pakietów

Tabela 12 Zależność natężenia danych i prawdopodobieństwa ataku (UDP Domain)

Natężenie danych [KB/s]	Prawdopodobieństwo
0,09	0,3

0,07	0,31
0,19	0,41
0,6	0,3
0,75	0,2
1,09	0,4
1,5	0,6
2,5	0,9
4,1	0,98
5,44	0,96
6,17	0,96

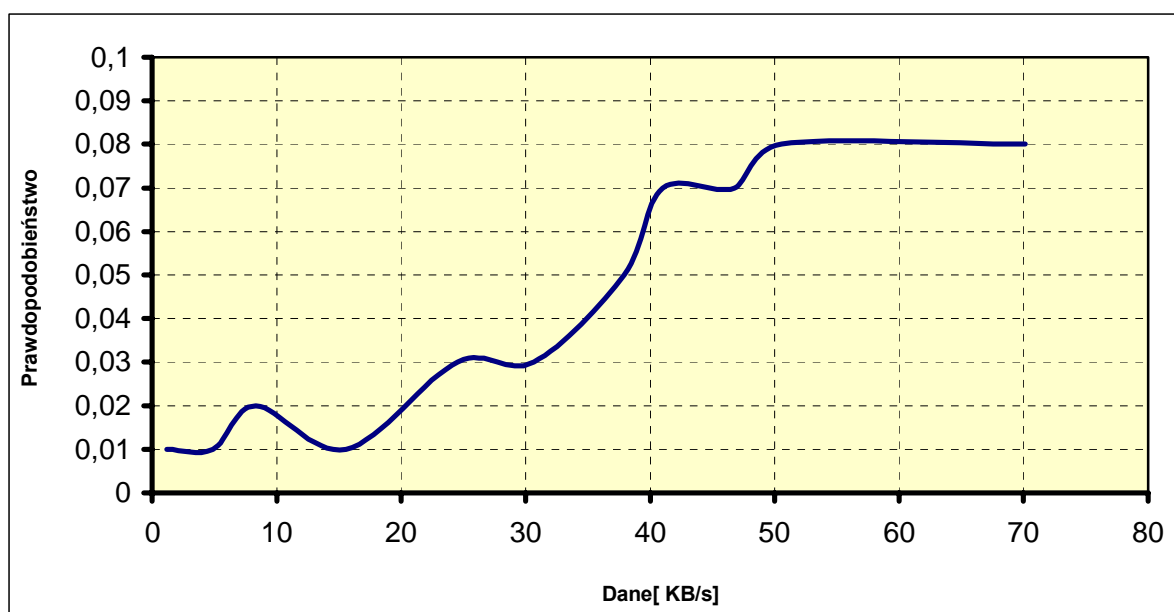


Rysunek 52 Test skuteczności – UDP Domain – rosnąca wielkość pakietów

Tabela 13 Zależność natężenia danych i prawdopodobieństwa ataku (TCP SYN)

Natężenie danych	Prawdopodobieństwo
------------------	--------------------

[KB/s]	
1,2	0,01
4,89	0,01
8,32	0,02
15,56	0,01
24,54	0,03
30,67	0,03
37,91	0,05
41,02	0,07
46,74	0,07
50,58	0,08
70,11	0,08



Rysunek 53 Test skuteczności – TCP SYN – rosnąca wielkość pakietów

Powyższe testy miały dać odpowiedź, jak system BIDS ocenia i klasyfikuje zwiększony ruch. Pierwsze trzy wykresy przedstawiają zmianę prawdopodobieństwa w zależności od liczby pakietów. Na wszystkich z nich można zauważyć, że przy małych wartościach liczby pakietów (ok. 10-12) gwałtownie rośnie prawdopodobieństwo, osiągając szybko poziom 0,8-0,9. Wpływ na takie zachowanie systemu mają dwa czynniki. Po pierwsze średnia ilość pakietów zawarta w pliku `net_stat.conf` dla protokołów ICMP i UDP wynosi około 10-13. W tym właśnie przedziale zmienna przekazywana do sieci Bayesa (ilość pakietów/s) zmienia się z L2 na L3. Drugim czynnikiem, jest natomiast zawartość danych uczących. Dane oznaczone jako normalne miały wspomniany czynnik w większości ustawiony na wartości L0..L2, natomiast dane uczące ataków L3...L5. Ten gwałtowny wzrost nie obniża skuteczności systemu, o ile będzie on występował przy odpowiednich przedziałach. Administrator ma możliwość zmiany wartości w pliku `net_stat.conf`.

W drugiej fazie tych testów, liczba pakietów była ustalana na wartość średnią dla danego protokołu, a zmianie ulegała ich wielkość, czyli wzrastała ilość danych docierających do serwera testowego. W przypadku pierwszych dwóch ataków (ICMP Redirect i UDP Domain Flood) wzrost natężenia danych pociągał za sobą wzrost prawdopodobieństwa. Można było zaobserwować, że wzrost ten nie jest jednostajny, są wartości, dla których prawdopodobieństwo powinno rosnać, a mimo to maleje. Przyczyną tego zachowania jest nie do końca dobre ustalenie tablic prawdopodobieństw w sieci Bayesa, które nastąpiło w wyniku samodzielnego uczenia się. Trzeci z wykresów, dotyczący ataku TCP SYN Flood, znacząco różni się od pozostałych dwóch. Widać na nim, że wzrost natężenia danych prawie w ogóle nie wpływa na wzrost prawdopodobieństwa. Przy maksymalnej wartości, prawdopodobieństwo wynosi zaledwie 8%. Jest to jak najbardziej prawidłowa wartość, gdyż w tego rodzaju ataku, agresorowi zależy na przesłaniu jak największej ilości pakietów i ich rozmiar powinien być jak najmniejszy.

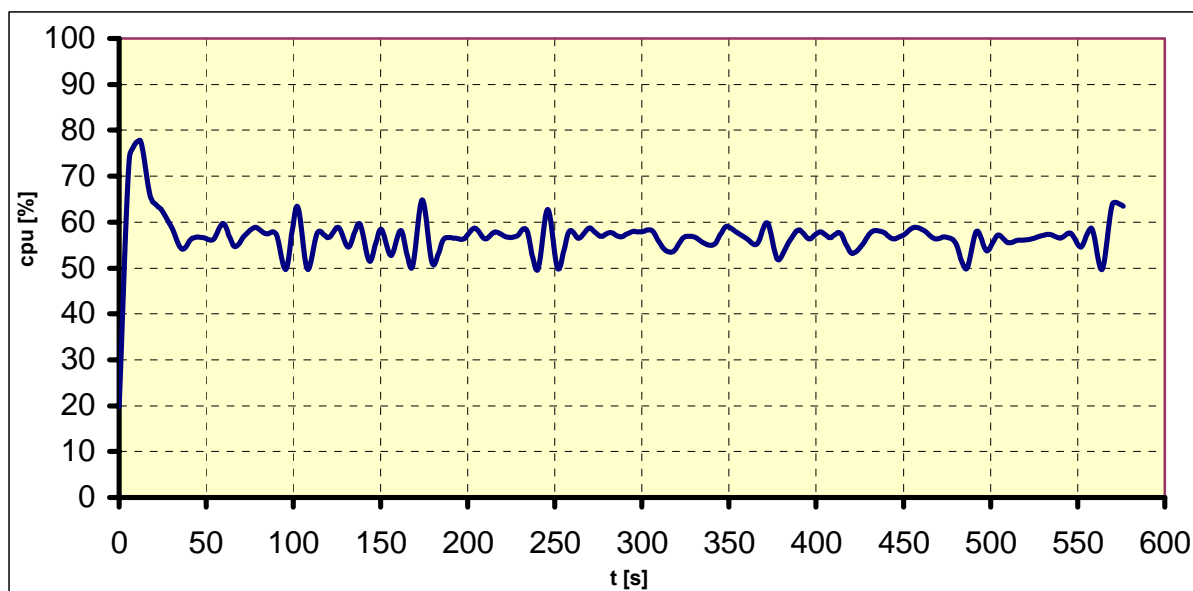
Testy wykazały, że pomimo pewnych drobnych niedociągnięć system BIDS dobrze rozpoznaje ataki i może być skutecznym i przydatnym narzędziem do walki z agresorami. Uzyskanie lepszych wyników wiązałoby się prawdopodobnie z udoskonaleniem procesu uczenia (bardziej reprezentatywne próbki) oraz zmodyfikowaniem sieci Bayesa.

### **8.4.3. Testy wydajnościowe**

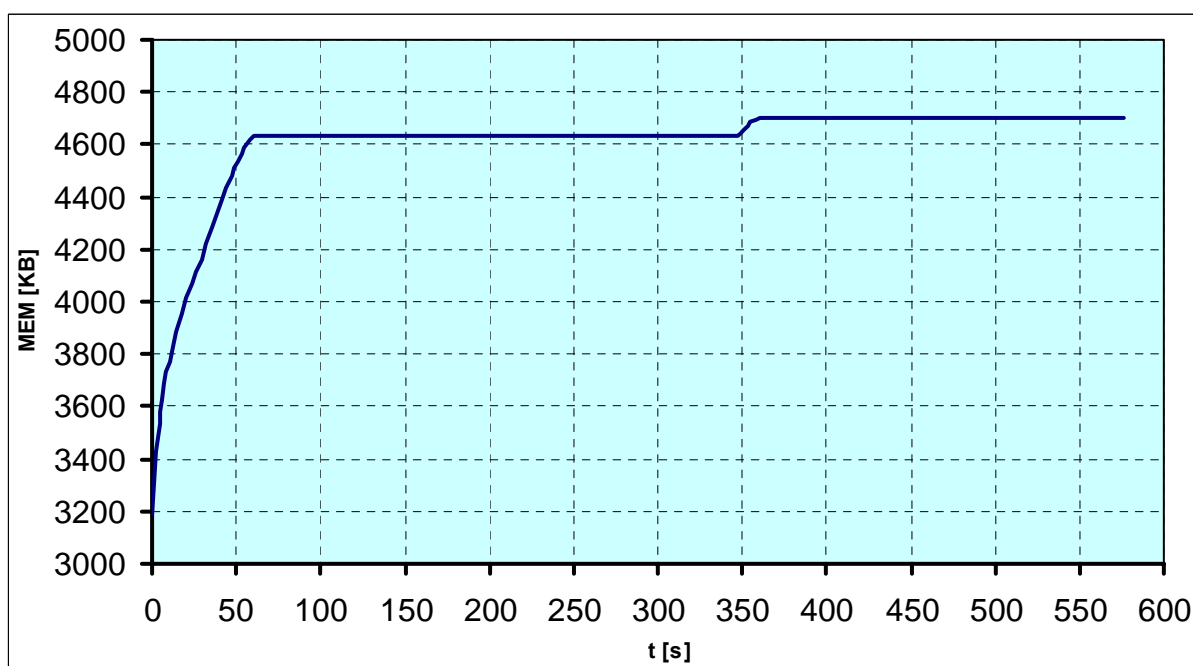
Celem tych testów było zmierzenie obciążenia systemu, jakie powoduje system BIDS podczas największej powodzi, jaką uda się zasymulować. W tym teście będą badane dwa kluczowe parametry:

- Ilość zużytej pamięci [w kB] - CPU.
- Zajętość czasu procesora [%] - MEM.

Działanie innych usług na testowanym serwerze została ograniczona do minimum, aby nie mogły one wpłynąć na wyniki. W tym teście serwer będzie kolejno zalewany pakietami: ICMP, UDP, TCP na przypadkowe usługi z przypadkowym adresem źródłowym oraz portem źródłowym. Czas pomiaru dla każdego z protokołów do 10 minut każdy. Wykresy obciążenia procesora i zużycia pamięci dotyczą typu procesu systemowego BIDS.

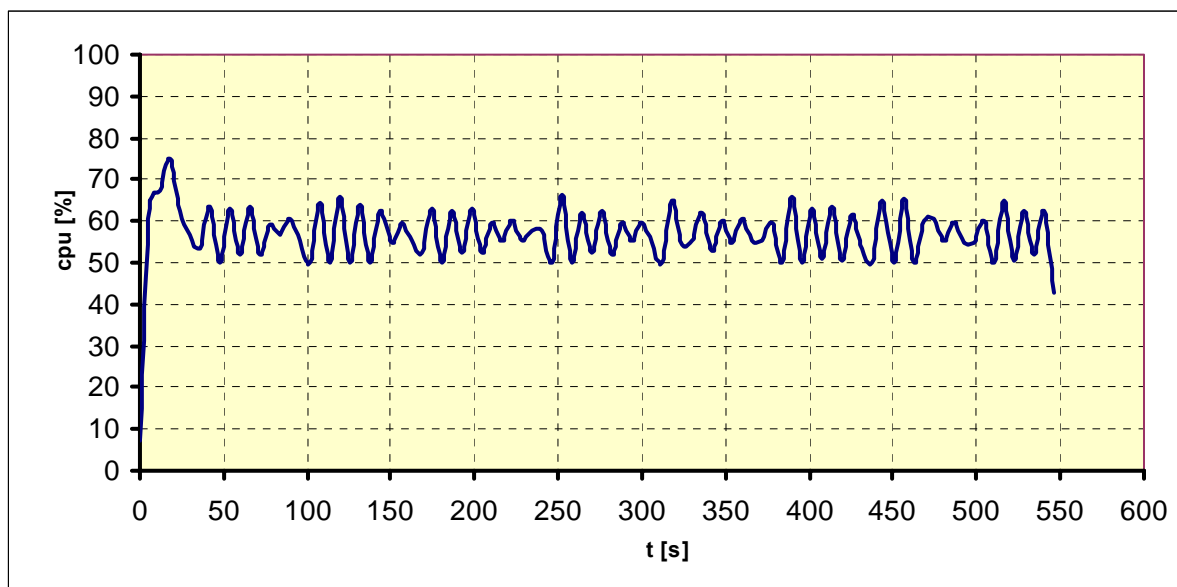


Rysunek 54 Testy obciążeniowe – ICMP -CPU

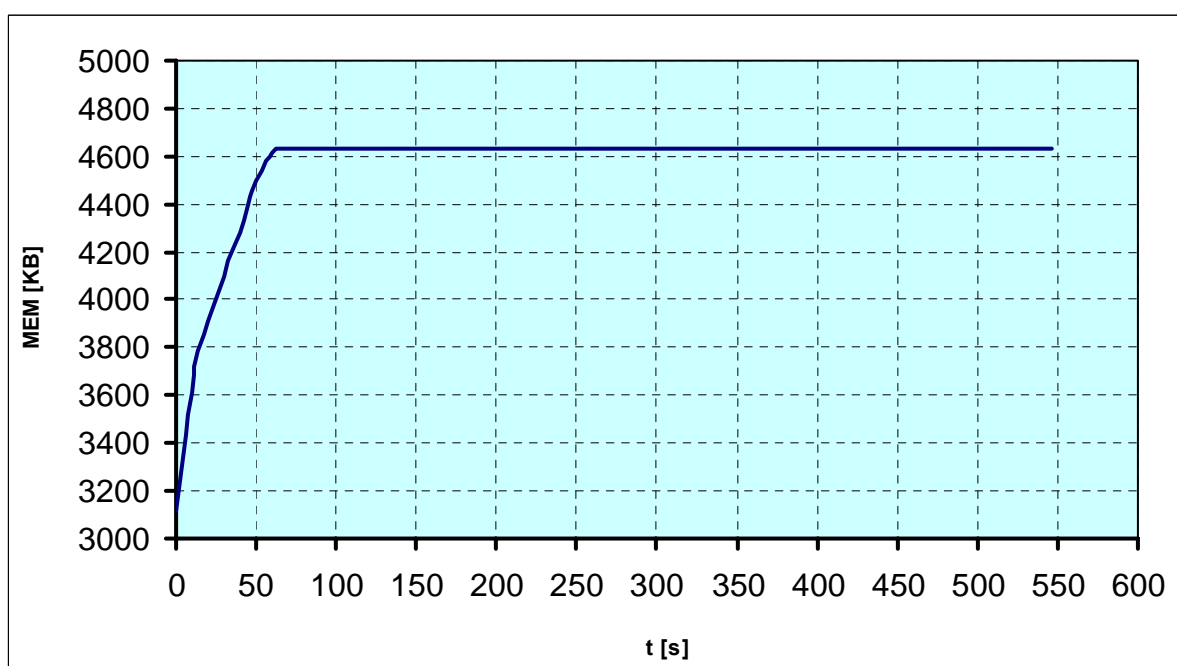


Rysunek 55 Testy obciążeniowe – ICMP -MEM

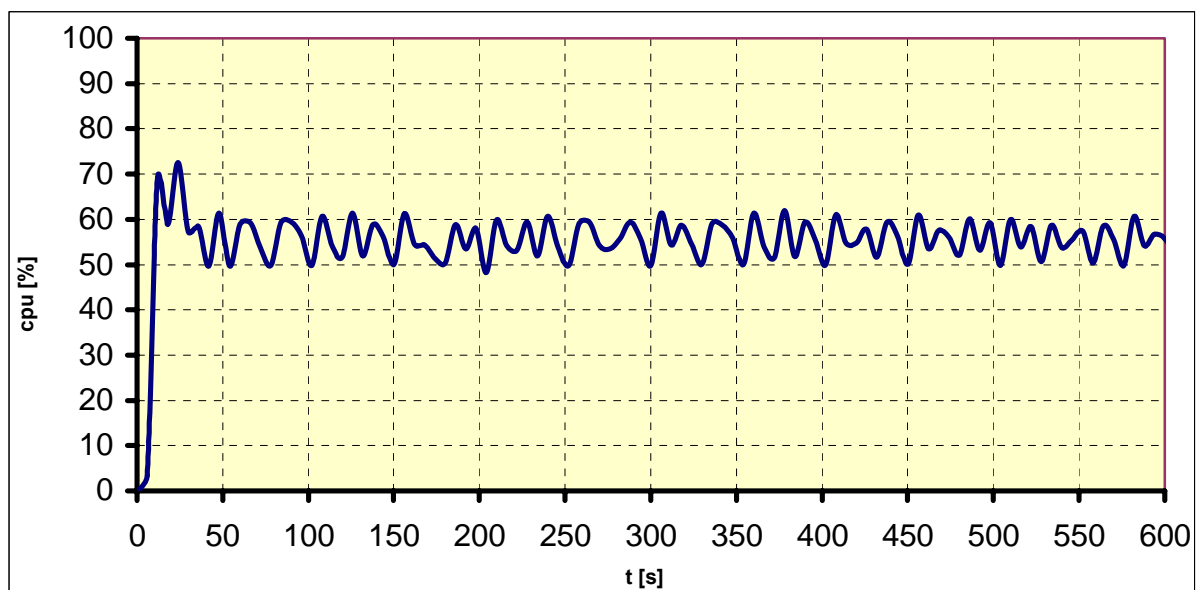




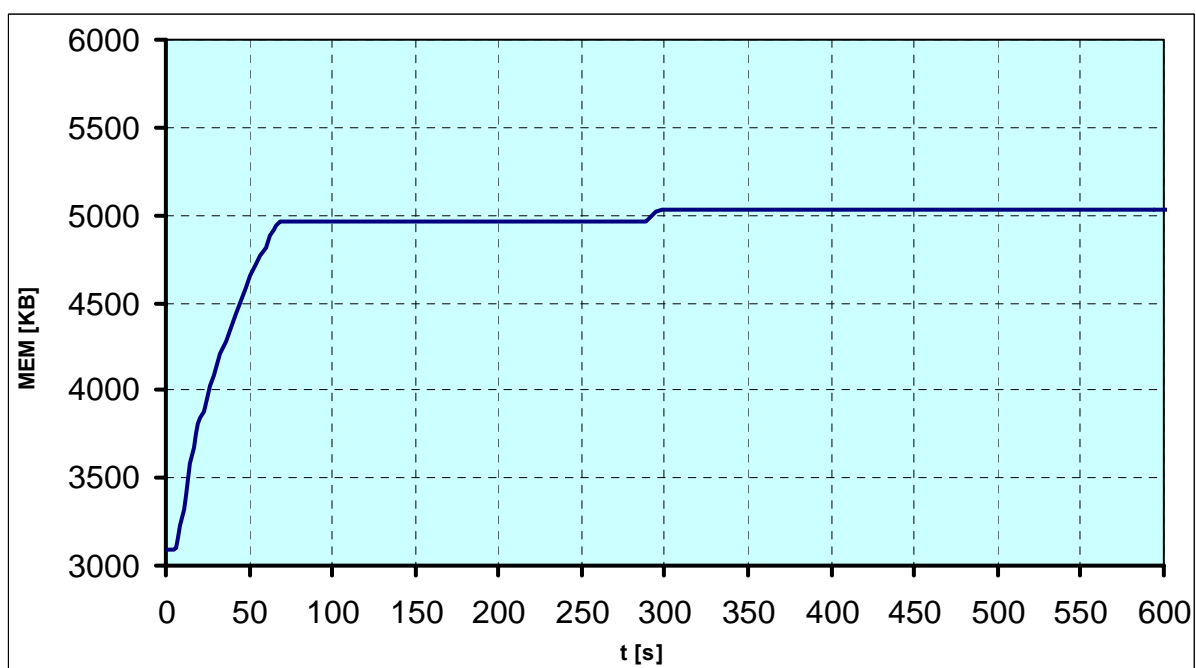
Rysunek 56 Testy obciążeniowe – UDP -CPU



Rysunek 57 Testy obciążeniowe – UDP -MEM



Rysunek 58 Testy obciążeniowe – TCP - CPU



Rysunek 59 Testy obciążeniowe – TCP -MEM

Tabela 14 Zestawienie obciążenia systemu

	ICMP	UDP	TCP
Czas pomiaru	576 s	546 s	711 s
Średnia ilość pakietów	571,77 p/s	420,23 p/s	370,72 p/s

Wartość średnia CPU	56,78 %	57,01 %	54,99 %
Wartość minimalna CPU	0 %	0 %	0 %
Wartość maksymalna CPU	77,88 %	74,86 %	75,96 %
Wartość średnia MEM	4603,05 KB	4561,89 KB	4899,42 KB
Wartość minimalna MEM	3072 KB	3084 KB	3088 KB
Wartość maksymalna MEM	4700 KB	4628 KB	5028 KB

Warunki przeprowadzonego testu miały na celu wywołanie maksymalnego obciążenia systemu. W sytuacji, kiedy każdy z pakietu wysłanego do serwera miał losowy numer IP i port (źródłowy), BIDS był zmuszony do przydzielenia każdemu z nich pojedynczego rekordu w pamięci. Dlatego we wszystkich przypadkach w ciągu pierwszych 60 s następuje gwałtowny wzrost wykorzystanej pamięci. Po 60 s (czas życia rekordu), pierwsze alokowane rekordy straciły swoją wartość i zostały nadpisane przez nowe dane. Dynamika przydziału nowych bloków spadła praktycznie do zera i utrzymała się do końca testu. Ilość zaalokowanej pamięci wynosiła maksymalnie ok 2MB (5028 KB - 3088 KB = 1940 KB) w przypadku protokołu TCP i nieznacznie mniej w przypadku pozostałych (UDP, ICMP). Wynika to z prostej przyczyny, że pojedynczy rekord dla protokołu TCP zajmuje o kilkadziesiąt bajtów więcej niż rekordy dla ICMP i UDP.

Inaczej przedstawiało się wykorzystanie procesora. We wszystkich przypadkach, z początkowego zera (brak ataku), nastąpił gwałtowny wzrost i osiągnięcie maksymalnej wartości w ciągu pierwszych 10 s – ICMP 77,88%. Następnie wykorzystanie CPU spadło nieznacznie o około 15-20%. W dalszych fazach testu w przypadku wszystkich protokołów wykorzystanie procesora zmieniało się cyklicznie w czasie (raz malało, raz wzrastało), nie wychodząc jednak z widełek 50%-60%. Można przypuszczać, że przyczynami takiego zachowania mogło być zmiana strategii przydzielania czasu procesora przez system albo konieczność synchronizowania jednej z funkcji wewnątrz wątku. Ustalenie dokładnej przyczyny wymaga zastosowania profilera i jest dość trudne do przeprowadzenia przy takim obciążeniu.

Generalnie system przeszedł pozytywnie testy wydajnościowe. Na plus jego można zaliczyć bardzo niewielkie wykorzystanie pamięci (do 5 MB), co w przypadku dzisiejszych programów jest dobrą wielkością. Nieco gorzej aplikacja radzi sobie z obciążaniem procesora – jest ono trochę za wysokie, ale mieszczące się w granicach rozsądku. Z pewnością pomogłoby dokładne przeanalizowanie całego kodu aplikacji i jego optymalizacja. Również zastąpienie sekwencyjnego przeszukiwania tablicy rekordów poprzez przeszukiwanie

uporządkowanego drzewa znacząco poprawiłoby wyniki. W dalszej kolejności można by rozważyć własną, wysoce zoptymalizowaną implementację sieci Bayesa.

## 9. Podsumowanie

W ostatnich latach ataki typu odmowa usługi (DoS, DDoS) stały się prawdziwą plagą. Ofiarami ich stają się największe firmy z branży nowych technologii: Yahoo, eBay, Amazon (2000), Microsoft (2003 i 2004), SCO (2004). Pomimo, że firmy te dysponują ogromnymi środkami, nie powstrzymały zmasowanych ataków na ich serwery. Problem z atakami typu DDoS polega na tym, że nie ma złotego środka zaradczego. Celem tej pracy było pełne umówienie podgrupy ataków DoS/DDoS występujących na warstwie sieciowej i transportowej modelu OSI, oraz zaimplementowanie systemu ostrzegania o atakach z wykorzystaniem sieci Bayesa.

Pierwsza część rozpoczęła się ogólną charakterystyką rodziny protokołów TCP/IP. Wiedza o nich jest niezbędna, aby móc zrozumieć anatomię większości ataków DoS/DDoS. W dalszej kolejności przedstawiono ogólną charakterystykę tego typu zagrożeń, fazy ich przeprowadzania oraz historia ataków. Ważną częścią opisu teoretycznego jest podział i klasyfikacja ataków DoS/DDoS. Wyróżniono 9 kryteriów klasyfikacji oraz 23 podgrupy. Następnie przeprowadzono podobną klasyfikację metod obrony (7 kryteriów klasyfikacji oraz 22 podkategorie). W kolejnym rozdziale został zawarty dokładny opis kilku popularnych ataków: powódź pakietów TCP z ustawioną flagą SYN (SYN Flood), pakiety z adresem docelowym i źródłowym ofiary (LAND), grupę ataków z wykorzystaniem powielaczy (SMURF, ICMP Redirect Flood, DNS Flood, HTTP Proxy), oraz dwa ataki wykorzystujące błędy w implementacji stosu TCP/IP (Ping of Death, Teardrop). Oprócz anatomii samych ataków zostały zaproponowane metody obrony przed nimi. Część teoretyczna kończy się opisem trzech narzędzi do przeprowadzania rozproszonych ataków DoS (Trinoo, TFN, Stacheldraht).

Druga część pracy zawiera opis i analizę systemu wykrywania ataków DoS/DDoS - BIDS (Bayesian Intrusion Detection System). System ten został napisany całkowicie od podstaw przez autora pracy. Aplikacja ta koncentruje się na wykrywaniu anomalii w natężeniu ruchu i jest szczególnie czuła na zwiększony ruch w wybranych protokołach. Celem projektowym było, aby system wykrywał nie tylko kilka określonych ataków, ale również potrafił wykryć te nieznane. Poprzez odpowiednio dobrany proces uczenia sieci Bayesa cel został osiągnięty. Pewną obawą autora było, czy system BIDS będzie w stanie zanalizować w czasie rzeczywistym wszystkie dane podczas ataku. Testy jednak wykazały, że obciążenie procesora i wykorzystanie pamięci mieści się w bezpiecznych granicach. Testy skuteczności wykrywania ataków, również wypadły dobrze – system poprawnie wykrywa zwiększone natężenie przesyłanych pakietów i klasyfikuje rodzaj ataku. Wykorzystanie sieci Bayesa, jako silnika do analizy przechwyconych okazało się bardzo dobrym pomysłem. Ogromną ich zaletą jest szybkość (kluczowa parametr) oraz możliwość uczenia się poprzez pokazywanie odpowiednich przykładów zachowania sieci.

Wszystkie założone cele związane z niniejszą pracą magisterską zostały zrealizowane.

## 10. Bibliografia

- [1] Open Systems Interconnection (OSI) Protocols, Cisco Systems, Inc, 2003,  
[http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito\\_doc/osi\\_prot.htm](http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/osi_prot.htm)
- [2] Internet Protocol, University of Southern California, 1981,  
<http://www.ietf.org/rfc/rfc0791.txt>
- [3] The Recommendation for the IP Next Generation Protocol, S. Bradner (Harvard University), A. Mankin (ISI), 1995  
<http://www.faqs.org/rfcs/rfc1752.html>
- [4] User Datagram Protocol, J. Postel, ISI, 1980  
<http://www.faqs.org/rfcs/rfc768.html>
- [5] Internet Control Message Protocol, J. Postel, 1981  
<http://www.faqs.org/rfcs/rfc792.html>
- [6] Transmission Control Protocol, University of Southern California, 1981  
<http://www.faqs.org/rfcs/rfc793.html>
- [7] File Transfer Protocol, A. Bhushan, MIT Project MAC, 1971  
<http://www.faqs.org/rfcs/rfc114.html>
- [8] Email Bombing and Spamming, CERT Coordination Center, 1999-2002  
[http://www.cert.org/tech\\_tips/email\\_bombing\\_spamming.html](http://www.cert.org/tech_tips/email_bombing_spamming.html)
- [9] Ranking.PL, Gemius SA  
<http://www.ranking.pl/rank.php?stat=sysoperAL>
- [10] Computer Cops - A Brief History of Denial-of-Service Attacks, Brian Krebs, Staff Writer  
<http://www.computercops.biz/article3963.html>
- [11] The Spread of the Code-Red Worm (CRv2), David Moore, Colleen Shannon, 2003  
[http://www.caida.org/analysis/security/code-red/coderedv2\\_analysis.xml](http://www.caida.org/analysis/security/code-red/coderedv2_analysis.xml)
- [12] Warhol Worms: The Potential for Very Fast Internet Plagues, Nicholas C Weaver, 2002  
<http://www.cs.berkeley.edu/~nweaver/warhol.html>
- [13] CERT® Advisory CA-2001-26 Nimda Worm, CERT Coordination Center, 2001  
<http://www.cert.org/advisories/CA-2001-26.html>
- [14] CERT® Incident Note IN-2001-03, CERT Coordination Center, 2001  
[http://www.cert.org/incident\\_notes/IN-2001-03.html](http://www.cert.org/incident_notes/IN-2001-03.html)
- [15] The Morris Internet Worm, Andy Sudduth, Harvard, 1988  
<http://www.snowplow.org/tom/worm/worm.html>
- [16] CERT® Incident Note IN-2001-01, CERT Coordination Center, 2001  
[http://www.cert.org/incident\\_notes/IN-2001-01.html](http://www.cert.org/incident_notes/IN-2001-01.html)
- [17] Tripwire - Products - Tripwire for Servers, Tripwire, Inc., 2004  
<http://www.tripwire.com/products/servers/>
- [18] McAfee Security - Personal Firewall, Network Associates, Inc., 2004  
[http://www.mcafee.com/myapps/firewall/ov\\_firewall.asp](http://www.mcafee.com/myapps/firewall/ov_firewall.asp)

- [19] McAfee Security – VirusScan, Network Associates, Inc., 2004  
<http://www.mcafee.com/myapps/vso/default.asp>
- [20] Cisco - Cisco Intrusion Detection, Cisco Systems, Inc, 2004  
<http://www.cisco.com/warp/public/cc/pd/sqsw/sqidsz/>
- [21] A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms, Jelena Mirkovic, Janice Martin, Peter Reiher, University of California  
[http://lasr.cs.ucla.edu/ddos/ucla\\_tech\\_report\\_020018.pdf](http://lasr.cs.ucla.edu/ddos/ucla_tech_report_020018.pdf)
- [22] Security Protocol Design via Authentication Tests, Joshua D. Guttman, The MITRE Corporation  
[http://www.ccs.neu.edu/home/guttman/at\\_design.pdf](http://www.ccs.neu.edu/home/guttman/at_design.pdf)
- [23] Check Point FireWall-1, Check Point Software Technologies Ltd., 1997  
<http://www.checkpoint.com/press/1996/synattack.html>
- [24] ICMP Traceback Messages, Steve Bellovin (AT&T Labs Research), Marcus Leech, Tom Taylor (Nortel Networks), 2003  
<http://www.ietf.org/proceedings/03mar/I-D/draft-ietf-itrace-04.txt>
- [25] Implementing Pushback: Router-Based Defense Against DDoS Attacks, John Ioannidis, Steven M. Bellovin, AT&T Labs Research  
<http://www.research.att.com/~smb/papers/ddos-lacc.pdf>
- [26] FBI Press Room - Press Release - 2000 – Mafiaboy  
<http://www.fbi.gov/pressrel/pressrel00/mafia080700.htm>
- [27] 'Mafiaboy' will be sentenced in April, FBI National Press Office, 2000  
<http://www.nwfusion.com/news/2001/0122sentence.html>
- [28] NetBouncer: Client-legitimacy-based High-performance DDoS Filtering, Roshan Thomas, Brian Mark, Tommy Johnson, James Croall, Network Associates Laboratories, Dept. of Electrical and Computer Engineering  
[http://www.lasr.cs.ucla.edu/classes/239\\_1.spring03/papers/netbouncer.pdf](http://www.lasr.cs.ucla.edu/classes/239_1.spring03/papers/netbouncer.pdf)
- [29] Strategies to Protect Against Distributed Denial of Service (DDoS) Attacks, Cisco Systems, Inc, 2003  
<http://www.cisco.com/warp/public/707/newsflash.html>
- [30] Opis jądra Linux-a, jego instalacji, Brian Ward, Bartosz Maruszewski, 1999  
<http://www.jtz.org.pl/Html/Kernel-HOWTO.pl.html>
- [31] SYN cookies, D. J. Bernstein  
<http://cr.yp.to/syncookies.html>
- [32] How To: Harden the TCP/IP Stack, Microsoft Corporation, 2004  
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/HTHardTCP.asp>
- [33] Land Attack, m3lt, 1997  
<http://www.weltregierung.de/security/roc/DoS/land.c>
- [34] n e t s c a n . o r g, 1999  
<http://www.netscan.org/>
- [35] Phenoelit Advisory, FX, FtR, kim0, 2002  
<http://www.phenoelit.de/stuff/CiscoICMP.txt>

- [36] Echo Protocol, J. Postel, ISI, 1983  
<http://www.ietf.org/rfc/rfc862.txt>
- [37] Character Generator Protocol, J. Postel, ISI, 1983  
<http://www.ietf.org/rfc/rfc864.txt>
- [38] Ping of Death, Malachi Kenney, 1996  
<http://www.insecure.org/sploits/ping-o-death.html>
- [39] The DoS Project's "trino" distributed denial of service attack tool, David Dittrich, University of Washington, 1999  
<http://staff.washington.edu/dittrich/misc/trino.analysis>
- [40] The "Tribe Flood Network" distributed denial of service attack tool, David Dittrich, University of Washington, 1999  
<http://staff.washington.edu/dittrich/misc/tfn.analysis>
- [41] Telnet Protocol Specification, J. Postel, J. Reynolds, ISI, 1983  
<http://www.faqs.org/rfcs/rfc854.html>
- [42] Hypertext Transfer Protocol -- HTTP/1.1, R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, 1999  
<http://www.faqs.org/rfcs/rfc2616.html>
- [43] Network Time Protocol, D.L. Mills, M/A-COM Linkabit, 1985  
<http://www.faqs.org/rfcs/rfc958.html>
- [44] Denial of Service (DoS) Attack Resource Page, Ferguson, DTS, 2000  
<http://www.denialinfo.com/>
- [45] Internet Relay Chat Protocol, J. Oikarinen, D. Reed, 1993  
<http://www.faqs.org/rfcs/rfc1459.html>
- [46] CERT® Advisory CA-1995-01 IP Spoofing Attacks and Hijacked Terminal Connections, CERT Coordination Center, 1997  
<http://www.cert.org/advisories/CA-1995-01.html>
- [47] Teardrop Attack, G P R, 1997  
<http://www.attrition.org/security/denial/w/teardrop.dos.html>
- [48] The "stacheldraht" distributed denial of service attack tool, David Dittrich, University of Washington, 1999  
<http://staff.washington.edu/dittrich/misc/stacheldraht.analysis>
- [49] Tcpdump - dump traffic on a network  
[http://www.tcpdump.org/tcpdump\\_man.html](http://www.tcpdump.org/tcpdump_man.html)
- [50] MRTG: The Multi Router Traffic Grapher, Tobias Oetiker, Dave Rand, 2004  
<http://people.ee.ethz.ch/~oetiker/webtools/mrtg/>
- [51] Norsys Software Corp., 2004  
<http://www.norsys.com/>
- [52] Programming POSIX Threads, 2004  
<http://www.humanfactor.com/pthreads/>
- [53] What is Bayesian Learning, Radford Neal, 2004  
<http://www.faqs.org/faqs/ai-faq/neural-nets/part3/section-7.html>



- [54] HPING, Salvatore Sanfilippo, 2004  
<http://www.hping.org/>
- [55] Bayesian Learning  
<http://l2r.cs.uiuc.edu/~danr/Teaching/CS346-04/Lectures/10-LecBayes-NB4.pdf>
- [56] Bayesian Learning  
<http://www.andrew.cmu.edu/user/dgovinda/pdf/bayes.pdf>
- [57] Snort, 2004  
<http://www.snort.org/>
- [58] Bezpieczeństwo systemów komputerowych, Łukasz Bodzianowski  
<http://www.bsk.konin.lm.pl/wstep.html>

## 11. Spisy

### 11.1. Spis tabel

Tabela 1 Historia ataków DoS/DDoS [10].....	18
Tabela 2 Przepustowość łącza i liczbę pakietów SYN .....	35
Tabela 3 Rekomendowane wartości rejestru dla systemu Windows [32].....	39
Tabela 4 Opcje linii poleceń systemu BIDS .....	65
Tabela 5 Podstawowe parametry w pliku config.h .....	83
Tabela 6 Zaawansowane parametry w pliku config.h.....	84
Tabela 7 Format pliku net.stat.....	85
Tabela 8 Zależność natężenia pakietów i prawdopodobieństwa ataku (ICMP Redirect) .....	88
Tabela 9 Zależność natężenia pakietów i prawdopodobieństwa ataku (UDP Domain).....	89
Tabela 10 Zależność natężenia pakietów i prawdopodobieństwa ataku (TCP SYN) .....	90
Tabela 11 Zależność natężenia danych i prawdopodobieństwa ataku (ICMP Redirect) .....	91
Tabela 12 Zależność natężenia danych i prawdopodobieństwa ataku (UDP Domain).....	92
Tabela 13 Zależność natężenia danych i prawdopodobieństwa ataku (TCP SYN) .....	93
Tabela 14 Zestawienie obciążenia systemu .....	98

### 11.2. Spis rysunków

Rysunek 1 Stos protokołów TCP/IP a model OSI .....	6
Rysunek 2 Nagłówek protokołu IP .....	8
Rysunek 3 Nagłówek protokołu ICMP .....	10
Rysunek 4 Nagłówek protokołu UDP .....	11
Rysunek 5 Nagłówek protokołu TCP.....	12
Rysunek 6 Nawiązywanie połączenia TCP – stan początkowy .....	13
Rysunek 7 Nawiązywanie połączenia TCP – pierwsza faza.....	13
Rysunek 8 Nawiązywanie połączenia TCP – druga faza .....	13
Rysunek 9 Nawiązywanie połączenia TCP – trzecia faza .....	14
Rysunek 10 Podział i klasyfikacja ataków typu DoS/DDoS [21].....	26
Rysunek 11 Podział i klasyfikacja metod obrony przed atakami typu DoS/DDoS [21].....	33
Rysunek 12 Przebieg ataku typu SYN Flood .....	34
Rysunek 13 Sposób działania mechanizmu SYNDefender .....	36
Rysunek 14 Przebieg ataku typu LAND .....	40
Rysunek 15 Sposób wykorzystania przykładowego powielacza .....	41
Rysunek 16 Przebieg ataku typu ICMP SMURF .....	43
Rysunek 17 Przykład odebrania pakietów ICMP z adresu rozgłoszeniowego .....	44
Rysunek 18 Działanie komunikatu ICMP Redirect .....	46
Rysunek 19 Przebieg ataku ICMP Redirect .....	47
Rysunek 20 Przebieg ataku z wykorzystaniem serwerów DNS .....	49
Rysunek 21 Przebieg ataku UDP Chargin.....	50
Rysunek 22 Przebieg ataku z wykorzystaniem HTTP Proxy .....	51
Rysunek 23 Prawidłowo pofragmentowany pakiet.....	53
Rysunek 24 Nieprawidłowy pakiet w ataku Teardrop .....	54
Rysunek 25 Schemat typowej sieci DDoS (Trinoo) .....	56
Rysunek 26 System BIDS – przechwytywanie pakietów .....	59
Rysunek 27 System BIDS - agregacja danych z otrzymanych pakietów .....	60

Rysunek 28 System BIDS – ładowanie danych z rekordów do sieci Bayesa .....	60
Rysunek 29 System BIDS – obliczanie prawdopodobieństwa ataku .....	61
Rysunek 30 Instalacja BIDS na bramie sieciowej .....	62
Rysunek 31 Instalacja BIDS wewnątrz sieci lokalnej .....	63
Rysunek 32 Instalacja BIDS na jednym z serwerów .....	64
Rysunek 33 Przykładowy wykres ilości odebranych i wysłanych danych (okres doby) .....	67
Rysunek 34 Przykładowy wykres ilości odebranych i wysłanych danych (okres tygodnia) ...	67
Rysunek 35 Przykładowy wykres ilości odebranych i wysłanych danych (okres miesiąca) ...	68
Rysunek 36 Przykładowy wykres ilości odebranych i wysłanych danych (okres roku) .....	68
Rysunek 37 Moduły systemu BIDS i kierunki przepływu danych .....	70
Rysunek 38 Schemat postępowania przy uogólnianiu danych .....	72
Rysunek 39 Graficzny moduł pakietu Netica (Windows) .....	74
Rysunek 40 Schemat sieci Bayesa .....	76
Rysunek 41 Wykres funkcji dyskretyzującej 1 .....	77
Rysunek 42 Wykres funkcji dyskretyzującej 2 .....	78
Rysunek 43 Przykładowy rozkład prawdopodobieństw w węźle sieci Bayesa .....	79
Rysunek 44 Schemat sieci podczas pierwszej fazy uczenia .....	80
Rysunek 45 Schemat sieci podczas drugiej fazy uczenia .....	81
Rysunek 46 Opcja uczenia pakietu Netica .....	82
Rysunek 47 Schemat środowiska testowego .....	87
Rysunek 48 Test skuteczności – ICMP Redirect – rosnąca liczba pakietów .....	89
Rysunek 49 Test skuteczności – UDP Domain – rosnąca liczba pakietów .....	90
Rysunek 50 Test skuteczności – TCP SYN – rosnąca liczba pakietów .....	91
Rysunek 51 Test skuteczności – ICMP Redirect – rosnąca wielkość pakietów .....	92
Rysunek 52 Test skuteczności – UDP Domain – rosnąca wielkość pakietów .....	93
Rysunek 53 Test skuteczności – TCP SYN – rosnąca wielkość pakietów .....	94
Rysunek 54 Testy obciążeniowe – ICMP -CPU .....	96
Rysunek 55 Testy obciążeniowe – ICMP -MEM .....	96
Rysunek 56 Testy obciążeniowe – UDP -CPU .....	97
Rysunek 57 Testy obciążeniowe – UDP -MEM .....	97
Rysunek 58 Testy obciążeniowe – TCP - CPU .....	98
Rysunek 59 Testy obciążeniowe – TCP -MEM .....	98

## 12. Dodatki

### 12.1. Przykładowe dane – statystyka sieci

UDP:1087207844.930300:0:0.074219:1  
UDP:1087207844.930300:1:0.128906:1  
TCP:1087207844.930300:0:1.243164:15  
TCP:1087207844.930300:1:0.597656:8  
TCP:1087207844.930300:0:0.078125:2  
TCP:1087207844.930300:0:0.281250:6  
TCP:1087207844.930300:1:0.365234:7  
TCP:1087207844.930300:0:0.449219:5  
TCP:1087207844.930300:0:0.199219:5  
TCP:1087207844.930300:0:0.367188:8  
TCP:1087207844.930300:1:17.578125:12  
TCP:1087207844.930300:0:0.421875:2  
TCP:1087207844.930300:0:0.946289:8  
TCP:1087207844.930300:1:0.406250:2  
TCP:1087207844.930300:1:6.140625:12  
TCP:1087207844.930300:0:0.373047:3  
TCP:1087207844.930300:1:0.085938:2  
TCP:1087207844.930300:0:7.402344:7  
TCP:1087207844.930300:1:0.234375:6  
TCP:1087207844.930300:1:0.042969:1  
TCP:1087207844.930300:0:0.078125:2  
TCP:1087207844.930300:1:30.667969:22  
TCP:1087207844.930300:0:0.546875:8  
TCP:1087207844.930300:1:0.535156:8  
TCP:1087207844.930300:0:0.101562:2  
TCP:1087207844.930300:1:0.050781:1  
TCP:1087207844.930300:0:0.046875:1

TCP:1087207844.930300:1:0.039062:1  
TCP:1087207844.930300:0:0.078125:2  
UDP:1087207847.030190:1:0.080078:1  
UDP:1087207847.030190:0:0.226562:1  
TCP:1087207847.030190:0:0.507812:13  
TCP:1087207847.030190:0:0.410156:2  
TCP:1087207847.030190:0:1.039062:24  
TCP:1087207847.030190:1:55.664062:38  
TCP:1087207847.030190:0:0.806641:3  
TCP:1087207847.030190:0:1.214844:7  
TCP:1087207847.030190:1:0.645508:4  
TCP:1087207847.030190:1:4.492188:9  
TCP:1087207847.030190:0:0.273438:7  
TCP:1087207847.030190:1:30.926758:22  
TCP:1087207847.030190:0:6.770508:5  
TCP:1087207847.030190:1:0.195312:5  
TCP:1087207847.030190:1:0.648438:2  
TCP:1087207847.030190:1:34.312500:24  
TCP:1087207847.030190:0:0.820312:12  
TCP:1087207847.030190:1:0.802734:12  
TCP:1087207847.030190:0:0.773438:4  
TCP:1087207847.030190:1:0.395508:3  
TCP:1087207847.030190:0:0.046875:1  
TCP:1087207847.030190:1:0.039062:1  
TCP:1087207847.030190:1:0.039062:1  
TCP:1087207847.030190:1:0.039062:1  
TCP:1087207847.030190:0:0.039062:1  
TCP:1087207847.030190:0:0.039062:1  
TCP:1087207847.030190:0:0.454102:3  
TCP:1087207847.030190:1:1.210938:4  
TCP:1087207847.030190:1:0.050781:1

TCP:1087207847.030190:0:0.050781:1

TCP:1087207847.030190:0:0.762695:2

TCP:1087207847.030190:1:7.226562:6

TCP:1087207849.130330:0:0.531250:3

## 12.2. Przykładowe dane – plik CASE

```
// ~-      E-1]-  
>[CAS      >~
```

IDnum	apsps	apcps	fcf	srv	atsps	atrps	atfps	ataps	atups	atpps	attack_type
1	L0	L0	false	tcp_other	L0	L0	L0	L2	L0	L2	normal
2	L0	L0	false	tcp_other	L0	L0	L0	L2	L0	L2	normal
3	L0	L0	false	udp_other	L0	L0	L0	L0	L0	L0	normal
4	L0	L1	false	udp_other	L0	L0	L0	L0	L0	L0	normal
5	L0	L0	false	udp_other	L0	L0	L0	L0	L0	L0	normal
6	L1	L1	false	udp_other	L0	L0	L0	L0	L0	L0	normal
7	L0	L0	false	udp_other	L0	L0	L0	L0	L0	L0	normal
8	L1	L0	false	udp_other	L0	L0	L0	L0	L0	L0	normal
9	L0	L0	false	udp_other	L0	L0	L0	L0	L0	L0	normal
10	L0	L0	false	udp_other	L0	L0	L0	L0	L0	L0	normal
11	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
12	L0	L1	false	tcp_http	L0	L0	L0	L4	L0	L0	normal
13	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
14	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
15	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
16	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
17	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L2	normal
18	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L5	normal
19	L0	L0	false	tcp_other	L0	L0	L0	L5	L0	L5	normal
20	L0	L0	false	tcp_other	L0	L0	L0	L5	L0	L5	normal
21	L0	L0	false	tcp_http	L0	L0	L0	L4	L0	L0	normal
22	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
23	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
24	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
25	L0	L0	false	tcp_http	L0	L0	L2	L2	L0	L0	normal
26	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
27	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
28	L0	L0	false	tcp_http	L0	L0	L0	L2	L0	L0	normal
29	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
30	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L3	normal
31	L0	L0	false	tcp_http	L0	L0	L0	L4	L0	L0	normal
32	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L1	normal
33	L0	L0	false	tcp_pop3	L0	L0	L0	L4	L0	L3	normal
34	L0	L0	false	tcp_pop3	L0	L0	L0	L4	L0	L3	normal
35	L0	L0	false	tcp_pop3	L0	L0	L0	L5	L0	L3	normal
36	L0	L0	false	tcp_pop3	L0	L0	L0	L5	L0	L3	normal
37	L0	L1	false	udp_other	L0	L0	L0	L0	L0	L0	normal
38	L2	L1	false	udp_other	L0	L0	L0	L0	L0	L0	normal

39	L0	L0	false	udp_other	L0	L0	L0	L0	L0	L0	normal
40	L1	L1	false	udp_other	L0	L0	L0	L0	L0	L0	normal
41	L0	L1	false	tcp_other	L0	L0	L0	L5	L0	L2	normal
42	L0	L1	false	tcp_other	L0	L0	L0	L5	L0	L2	normal
43	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
44	L0	L2	false	tcp_http	L0	L0	L0	L4	L0	L0	normal
45	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
46	L1	L2	false	tcp_http	L0	L0	L0	L5	L0	L1	normal
47	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
48	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
49	L0	L1	false	tcp_other	L0	L0	L0	L5	L0	L4	normal
50	L0	L1	false	tcp_other	L0	L0	L0	L5	L0	L4	normal
51	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
52	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
53	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
54	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
55	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
56	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L1	normal
57	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
58	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L1	normal
59	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
60	L0	L0	false	tcp_pop3	L0	L0	L2	L5	L0	L0	normal
61	L0	L0	false	tcp_pop3	L0	L0	L2	L5	L0	L0	normal
62	L0	L0	false	tcp_pop3	L0	L0	L0	L2	L0	L0	normal
63	L0	L0	false	tcp_pop3	L0	L0	L0	L2	L0	L0	normal
64	L0	L0	false	tcp_http	L0	L0	L2	L2	L0	L0	normal
65	L0	L0	false	tcp_http	L0	L2	L0	L2	L0	L0	normal
66	L0	L0	false	tcp_http	L0	L5	L0	L0	L0	L0	normal
67	L0	L1	false	tcp_http	L0	L0	L0	L4	L0	L2	normal
68	L0	L1	false	tcp_http	L1	L0	L0	L5	L0	L2	normal
69	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
70	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L1	normal
71	L0	L0	false	tcp_http	L0	L0	L2	L5	L0	L0	normal
72	L0	L0	false	tcp_http	L0	L0	L2	L5	L0	L0	normal
73	L0	L0	false	tcp_http	L1	L0	L0	L3	L0	L1	normal
74	L0	L0	false	tcp_http	L1	L0	L0	L5	L0	L2	normal
75	L0	L1	false	tcp_http	L0	L0	L0	L4	L0	L0	normal
76	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L1	normal
77	L0	L0	false	tcp_http	L0	L2	L0	L0	L0	L0	normal
78	L0	L0	false	tcp_http	L0	L0	L2	L2	L0	L0	normal
79	L0	L0	false	tcp_http	L0	L0	L0	L2	L0	L0	normal
80	L0	L0	false	tcp_http	L0	L0	L0	L3	L0	L0	normal
81	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L2	normal
82	L0	L1	false	tcp_other	L0	L0	L0	L5	L0	L1	normal
83	L0	L1	false	tcp_other	L0	L0	L0	L5	L0	L1	normal
84	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
85	L0	L2	false	tcp_http	L0	L0	L0	L4	L0	L0	normal
86	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
87	L1	L2	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
88	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
89	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
90	L0	L1	false	tcp_other	L0	L0	L0	L5	L0	L4	normal

91	L0	L1	false	tcp_other	L0	L0	L0	L5	L0	L4	normal
92	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
93	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L1	normal
94	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
95	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
96	L0	L1	false	tcp_http	L0	L0	L0	L4	L0	L0	normal
97	L0	L0	false	tcp_http	L2	L0	L0	L5	L0	L2	normal
98	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
99	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L1	normal
100	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
101	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
102	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
103	L0	L1	false	tcp_http	L0	L0	L0	L4	L0	L0	normal
104	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
105	L0	L1	false	tcp_http	L0	L0	L0	L4	L0	L1	normal
106	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L2	normal
107	L0	L1	false	tcp_http	L0	L0	L0	L4	L0	L0	normal
108	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L1	normal
109	L0	L0	false	tcp_smtp	L2	L0	L0	L2	L0	L0	normal
110	L0	L0	false	tcp_smtp	L2	L0	L0	L2	L0	L0	normal
111	L0	L0	false	tcp_smtp	L2	L0	L0	L5	L0	L2	normal
112	L0	L0	false	tcp_smtp	L2	L0	L0	L5	L0	L2	normal
113	L0	L0	false	tcp_http	L0	L2	L0	L0	L0	L0	normal
114	L0	L1	false	tcp_http	L0	L0	L0	L4	L0	L0	normal
115	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
116	L0	L0	false	tcp_http	L0	L0	L2	L2	L0	L0	normal
117	L0	L0	false	tcp_http	L0	L0	L2	L2	L0	L0	normal
118	L0	L0	false	tcp_http	L0	L0	L0	L2	L0	L0	normal
119	L0	L0	false	tcp_http	L0	L0	L0	L2	L0	L0	normal
120	L0	L0	false	tcp_smtp	L1	L0	L0	L3	L0	L2	normal
121	L0	L0	false	tcp_smtp	L1	L0	L0	L5	L0	L3	normal
122	L0	L0	false	tcp_other	L0	L0	L0	L5	L0	L1	normal
123	L0	L0	false	tcp_other	L0	L0	L0	L5	L0	L1	normal
124	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
125	L0	L2	false	tcp_http	L0	L0	L0	L4	L0	L0	normal
126	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
127	L1	L2	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
128	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
129	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
130	L0	L0	false	tcp_other	L0	L0	L0	L5	L0	L3	normal
131	L0	L0	false	tcp_other	L0	L0	L0	L5	L0	L3	normal
132	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
133	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L2	normal
134	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
135	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
136	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
137	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
138	L0	L1	false	tcp_http	L0	L1	L0	L3	L0	L0	normal
139	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L1	normal
140	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
141	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
142	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal



143	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
144	L1	L1	false	tcp_smtp	L0	L0	L0	L5	L0	L2	normal
145	L1	L2	false	tcp_smtp	L0	L0	L0	L4	L0	L2	normal
146	L0	L1	false	tcp_smtp	L0	L0	L0	L5	L0	L0	normal
147	L0	L1	false	tcp_smtp	L0	L0	L0	L5	L0	L0	normal
148	L0	L0	false	tcp_http	L0	L0	L2	L2	L0	L0	normal
149	L0	L0	false	tcp_http	L0	L0	L0	L2	L0	L0	normal
150	L0	L1	false	tcp_http	L0	L0	L0	L4	L0	L0	normal
151	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
152	L0	L0	false	tcp_http	L0	L0	L0	L4	L0	L0	normal
153	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L1	normal
154	L0	L0	false	udp_domain	L0	L0	L0	L0	L0	L0	normal
155	L0	L1	false	udp_domain	L0	L0	L0	L0	L0	L0	normal
156	L0	L0	false	udp_domain	L0	L0	L0	L0	L0	L0	normal
157	L0	L0	false	udp_domain	L0	L0	L0	L0	L0	L0	normal
158	L0	L0	false	udp_domain	L0	L0	L0	L0	L0	L0	normal
159	L0	L0	false	udp_domain	L0	L0	L0	L0	L0	L0	normal
160	L0	L1	false	tcp_other	L0	L0	L0	L5	L0	L0	normal
161	L0	L1	false	tcp_other	L0	L0	L0	L5	L0	L0	normal
162	L0	L0	false	tcp_http	L0	L0	L0	L2	L0	L0	normal
163	L0	L2	false	tcp_http	L0	L0	L0	L4	L0	L0	normal
164	L0	L2	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
165	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
166	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
167	L0	L1	false	tcp_other	L0	L0	L0	L5	L0	L4	normal
168	L0	L1	false	tcp_other	L0	L0	L0	L5	L0	L4	normal
169	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
170	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
171	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L1	normal
172	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
173	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L2	normal
174	L0	L0	false	tcp_http	L0	L0	L0	L4	L0	L0	normal
175	L0	L1	false	tcp_http	L0	L5	L0	L0	L0	L0	normal
176	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
177	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
178	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
179	L0	L1	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
180	L0	L1	false	tcp_smtp	L0	L0	L1	L5	L0	L3	normal
181	L1	L1	false	tcp_smtp	L0	L0	L0	L5	L0	L4	normal
182	L0	L0	false	tcp_smtp	L0	L0	L1	L5	L0	L1	normal
183	L0	L1	false	tcp_smtp	L0	L0	L0	L5	L0	L0	normal
184	L0	L0	false	tcp_http	L0	L0	L2	L2	L0	L0	normal
185	L0	L0	false	tcp_http	L0	L3	L0	L1	L0	L0	normal
186	L1	L1	false	tcp_smtp	L0	L0	L0	L5	L0	L4	normal
187	L0	L1	false	tcp_smtp	L0	L0	L0	L5	L0	L0	normal
188	L0	L0	false	tcp_http	L0	L2	L0	L0	L0	L0	normal
189	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L0	normal
190	L0	L0	false	tcp_http	L0	L0	L0	L5	L0	L2	normal
191	L0	L0	false	tcp_http	L0	L0	L2	L5	L0	L0	normal
192	L0	L0	false	tcp_http	L0	L0	L2	L5	L0	L0	normal
193	L0	L0	false	tcp_http	L1	L0	L0	L3	L0	L1	normal
194	L0	L0	false	tcp_http	L1	L0	L0	L5	L0	L2	normal

195	L0	L0	false	tcp_http	L0	L0	L2	L2	L0	L0	normal
196	L0	L0	false	tcp_http	L2	L0	L0	L0	L0	L0	normal
197	L0	L0	false	tcp_http	L2	L0	L0	L2	L0	L0	normal

### 12.3. Struktury wykorzystywane w systemie

```
// Protokol ICMP

struct traffic_icmp
{
    in_addr_t saddr;
    in_addr_t daddr;
    traffic_dir dir;
    u_char icmp_type;
    unsigned int size;
    unsigned int count;
    struct traffic_icmp *ptr;
    int t_last_mod;
    int t_init;
    double t_analyse;
    pthread_mutex_t lock;
};

// Protokol UDP

struct traffic_udp
{
    in_addr_t saddr;
    u_short sport;
    in_addr_t daddr;
    u_short dport;
    traffic_dir dir;
    unsigned int size;
    unsigned int count;
    struct traffic_udp *ptr;
};
```

```

    int t_last_mod;

    int t_init;

    int t_analyse;

    pthread_mutex_t lock;
};

// Protokol TCP

struct traffic_tcp
{
    in_addr_t saddr;

    u_short sport;

    in_addr_t daddr;

    u_short dport;

    traffic_dir dir;

    unsigned int tcp_flags[MAX_TCP_FLAGS];

    unsigned int size;

    unsigned int count;

    struct traffic_tcp *ptr;

    int t_last_mod;

    int t_init;

    int t_analyse;

    pthread_mutex_t lock;
};

// Nieznany protokol

struct traffic_up
{
    in_addr_t saddr;

    in_addr_t daddr;

    traffic_dir dir;

    unsigned int size;

    unsigned int count;

    struct traffic_up *ptr;
};

```

```

    int t_last_mod;

    int t_init;

    int t_analyse;

    pthread_mutex_t lock;
};

```

## 12.4. Funkcje biblioteki PCAP

```

#include <pcap.h>

char errbuf[PCAP_ERRBUF_SIZE];

pcap_t *pcap_open_live(const char *device, int snaplen, int promisc, int
to_ms, char *errbuf);

pcap_t *pcap_open_dead(int linktype, int snaplen);

pcap_t *pcap_open_offline(const char *fname, char *errbuf);

pcap_dumper_t *pcap_dump_open(pcap_t *p, const char *fname);

int pcap_setnonblock(pcap_t *p, int nonblock, char *errbuf);

int pcap_getnonblock(pcap_t *p, char *errbuf);

int pcap_findalldevs(pcap_if_t **alldevsp, char *errbuf);

void pcap_freealldevs(pcap_if_t *alldevs);

char *pcap_lookupdev(char *errbuf);

int pcap_lookupnet(const char *device, bpf_u_int32 *netp, bpf_u_int32
*maskp, char *errbuf);

int pcap_dispatch(pcap_t *p, int cnt, pcap_handler callback, u_char *user);

int pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user);

void pcap_dump(u_char *user, struct pcap_pkthdr *h, u_char *sp);

int pcap_compile(pcap_t *p, struct bpf_program *fp, char *str, int
optimize, bpf_u_int32 netmask);

int pcap_setfilter(pcap_t *p, struct bpf_program *fp);

void pcap_freecode(struct bpf_program *);

const u_char *pcap_next(pcap_t *p, struct pcap_pkthdr *h);

int pcap_next_ex(pcap_t *p, struct pcap_pkthdr **pkt_header, const u_char
**pkt_data);

void pcap_breakloop(pcap_t *);

```

```
int pcap_datalink(pcap_t *p);
int pcap_list_datalinks(pcap_t *p, int **dlt_buf);
int pcap_set_datalink(pcap_t *p, int dlt);
int pcap_datalink_name_to_val(const char *name);
const char *pcap_datalink_val_to_name(int dlt);
const char *pcap_datalink_val_to_description(int dlt);
int pcap_snapshot(pcap_t *p);
int pcap_is_swapped(pcap_t *p);
int pcap_major_version(pcap_t *p);
int pcap_minor_version(pcap_t *p);
int pcap_stats(pcap_t *p, struct pcap_stat *ps);
FILE *pcap_file(pcap_t *p);
int pcap_fileno(pcap_t *p);
void pcap_perror(pcap_t *p, char *prefix);
char *pcap_geterr(pcap_t *p);
char *pcap_strerror(int error);
const char *pcap_lib_version(void);
void pcap_close(pcap_t *p);
int pcap_dump_flush(pcap_dumper_t *p);
FILE *pcap_dump_file(pcap_dumper_t *p);
void pcap_dump_close(pcap_dumper_t *p);
```