Praktyczna analiza sieci komputerowych z wykorzystaniem programu Wireshark

Mariusz Piwiński
Instytut Fizyki, Wydział Fizyki, Astronomii i Informatyki Stosowanej
Uniwersytet Mikołaja Kopernika w Toruniu
ul. Grudziądzka 5/7 87-100 Toruń
e-mail:Mariusz.Piwinski@fizyka.umk.pl

1. Wstęp

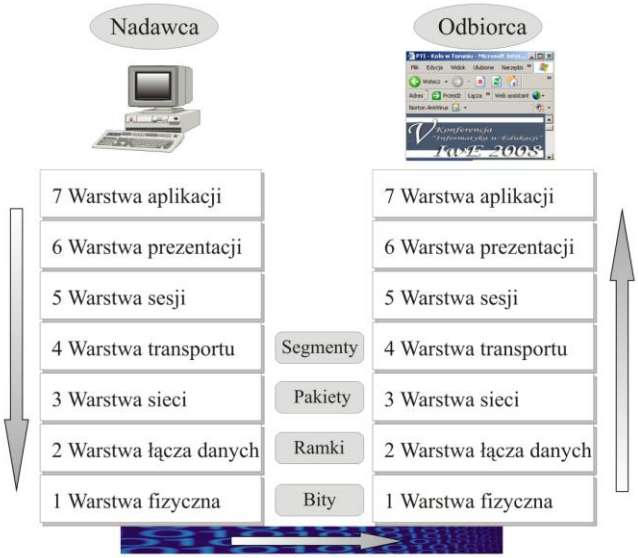
Życie w społeczeństwie informatycznym skutkuje wszechobecnością komputerów. Stały się one niezbędnym narzędziem wykorzystywanym w codziennej pracy zawodowej oraz w życiu prywatnym. Szybki rozwój technologiczny sprawił, że standardem stało się nie tylko posiadanie komputera osobistego, ale również jego podłączenie do Internetu. Niestety dostęp do tzw. *sieci*, który jest rzeczą naturalną w aglomeracjach miejskich, nie jest już tak oczywisty na terenach wiejskich. Możliwość załatwienia coraz większej ilości spraw *on-line,* w znaczący sposób pozwala na zaoszczędzenie czasu oraz dotarcia do usług, które nie są dostępne w pobliżu bezpośredniego miejsca zamieszkania. Zatem *on-line* możemy złożyć dokumenty na studia, założyć konto w banku, zrobić przelewy, a nawet zobaczyć co robi dziecko w przedszkolu. Oznacza to, iż przez sieci komputerowe przesyłamy bardzo wiele różnych informacji, które czasami mogą zostać wykorzystane bez naszej wiedzy przez osoby trzecie. Warto zastanowić się, przez chwilę czy aby na pewno wszystko co robimy jest bezpieczne i czy poczucie anonimowości, którą daje nam Internet jest uzasadnione.

2. Jak to działa, czyli model OSI i enkapsulacja

Poziom świadomości istnienia poszczególnych zagrożeń związany jest bezpośrednio ze stanem naszej wiedzy na dany temat, dlatego tak ważnym jest umiejętne uczenie zagadnień związanych z sieciami komputerowymi. Niestety zagadnienia te nie należą do najprostszych, a ciągły rozwój technologii sprawia, iż wdrażane są coraz nowsze rozwiązania. Należy zatem zacząć od najprostszych faktów związanych z działaniem sieci komputerowej. Po pierwsze musimy sobie uświadomić, iż podłączając komputer do sieci staje się on jednym z wielu tzw. hostów podłączonych do jakichś (często bliżej nieokreślonych) urządzeń sieciowych. Niezależnie od tego czy mamy do dyspozycji pojedynczy komputer, czy też sieć lokalną zawsze można wydzielić obszar uznawany za bezpieczny oraz resztę, na której działanie nie mamy bezpośredniego wpływu. W obszarze sieci, którą uznajemy za zaufaną mamy pełną kontrolę nad przepływającymi danymi oraz urządzeniami sieciowymi, co w praktyce oznacza, że osobiście sprawujemy nad nią nadzór, lub robi to zaufany administrator. Niestety w wielu przypadkach, aby skorzystać z usług dostępnych na serwerach, dane muszą opuścić bezpieczny obszar naszej sieci i przejść do strefy, nad którą nie mamy już żadnej kontroli. W efekcie oznacza to, iż różne osoby będą mogły wysyłane informacje przeglądać, gromadzić oraz co gorsza czasami również próbować je modyfikować.

Na początek zajmijmy się siecią, którą uznaliśmy za bezpieczną, która zazwyczaj jest utożsamiana z siecią lokalną LAN. Najczęstszą technologią wykorzystywaną na tym obszarze jest Ethernet określany również standardem IEEE 802.3 (co zresztą nie jest w pełni zgodne z prawdą). W celu lepszego zrozumienia poruszanych zagadnień warto przypomnieć tutaj warstwowy model odniesienia

OSI (*ang. Open System Interconnection*) określony w 1984 roku przez organizację ISO (*ang. International Organization for Standardization*). Składa się on z siedmiu ustawionych hierarchicznie warstw, z którymi można utożsamiać poszczególne procesy oraz realizujące je urządzenia. Takie podejście pozwala na podzielenie złożonych zagadnień na mniej skomplikowane elementy, którymi w praktyce jest znacznie łatwiej kierować. Prześledźmy zatem proces przesyłania danych na tle tego modelu (Rys.1).



Rys. 1 Przesyłanie informacji na tle modelu OSI.

Dla ułatwienia załóżmy, iż serwer WWW otrzymał od hosta żądanie przesłania pliku, który przedstawia np. program interesującej nas konferencji. Przesyłanie pliku rozpoczyna się od przygotowania danych w odpowiedniej formie, a następnie określenia momentu, w którym mają one zostać obsłużone przez proces umożliwiający ostatecznie wysłanie ich przez interfejs serwera. Wszystkie te elementy realizowane są w trzech najwyższych warstwach modelu OSI (warstwy 7, 6 i 5) operujących na strumieniu danych, który kierowany jest ostatecznie do warstwy transportu. Okazuje się jednak, że dane te stanowią bardzo duży zbiór informacji i nie mogą być zawarte w pojedynczym pakiecie. Maksymalną wielkość pakietu, który będzie przesyłany bez fragmentacji określa parametr MTU (*ang. Maximum Transmission Unit*). Standardowo dla sieci Ethernet wynosi on 1500 bajtów. W związku z powyższym dane muszą zostać podzielone na segmenty, które są kolejno numerowane, co umożliwi ich ponowne scalenie przez host docelowy. Opisywana transmisja związana jest z konkretnymi usługami oraz aplikacjami, dlatego też każdy z segmentów zostaje opatrzony dodatkowo numerami portów. Skoro korzystamy z protokołu HTTP, to port źródłowy otrzyma wartość 80. Port docelowy związany jest bezpośrednio z aplikacją uruchomioną na komputerze użytkownika, czyli w tym przypadku przeglądarką internetową, co oznacza, że np. może otrzymać numer 1049. Tak utworzone segmenty są

następnie przekazywane do warstwy sieci. Na jej poziomie, każdy z nich otrzymuje adres IP źródłowy (w naszym przypadku serwera WWW) i adres IP docelowy (hosta, który wysłał żądanie). Opakowany segment nosi nazwę pakietu i przesyłany jest do warstwy niższej, czyli warstwy łącza danych. Warstwa ta dokonuje dalszego procesu enkapsulacji (czyli opakowania danych) dodając między innymi adres MAC źródłowy (adres fizyczny interfejsu serwera WWW) oraz adres MAC docelowy. Zauważmy tutaj, iż w przypadku, gdy host docelowy znajduje się w naszej sieci to adresem MAC docelowym jest adres MAC interfejsu hosta docelowego, zaś w przeciwnym razie jest to adres fizyczny interfejsu bramy. Utworzone w ten sposób ramki przekazywane są do warstwy fizycznej, która odpowiada za wysłanie je poprzez istniejące medium zgodnie z kodowaniem określonym przez standard, w którym pracuje interfejs nadawcy (np. dla 100BaseTX będzie to MLT-3 wspierane dodatkowo przez 4B/5B). Wysłane dane są odbierane i przekazywane dalej przez różne urządzenia sieciowe, aż trafiają wreszcie do odbiorcy, gdzie następuje proces ich weryfikacji. Na poziomie warstwy fizycznej zamieniane są one na postać cyfrową, a następnie przekazywane do warstwy łącza danych. Na jej poziomie porównywany jest adres MAC docelowy umieszczony w ramce z faktycznym adresem MAC interfejsu. Jeżeli weryfikacja zostanie zakończona sukcesem, dane przekazywane są do warstwy wyższej, w przeciwnym przypadku są one odrzucane. Zakładając, że adres MAC był jednak prawidłowy, kolejnym krokiem realizowanym przez warstwę sieci, będzie porównanie docelowego adresu IP z faktycznym logicznym adresem interfejsu. Tutaj również przekazanie danych do warstwy wyższej nastąpi tylko w przypadku poprawnej weryfikacji. Ostatecznie wszystkie przesłane w ten sposób segmenty zostaną ponownie złożone w jedną całość na poziomie warstwy transportu, a zawarty numer portu docelowego pozwoli systemowi na jednoznaczne określenie, która z uruchomionych aplikacji jest prawidłowym odbiorcą danych. Oczywiście na hoście docelowym uruchomionych jest szereg procesów, a zatem warstwa sesji będzie odpowiedzialna za obsługę danych w odpowiednim momencie, a warstwy prezentacji i aplikacji umożliwią poprawne ich zinterpretowanie przez aplikację. Aby w znaczący sposób uprościć sytuację, opisując proces przesyłania danych skoncentrowałem się tylko i wyłącznie na transmisji z serwera do hosta. Trzeba mieć jednak świadomość, iż jest to zaledwie mały fragment tego co dzieje się w sieci.

3. Jak to wygląda w praktyce?

Po przeprowadzonej analizie mogą nasunąć się pierwsze pytania. Po pierwsze w pakietach mówimy o adresach IP, a w przeglądarkach używamy raczej nazw. Po drugie skąd nasz host zna adres MAC hosta docelowego? Jak widać w stosie protokołów TCP/IP oprócz tych dwóch podstawowych znajdują się również inne, których działania w większości przypadków przeciętny użytkownik nie zauważa. Mając podstawową wiedzę na temat enkapsulacji danych oraz modelu OSI trzeba powiedzieć, iż nie wszystko rozpoczyna się w warstwie aplikacji. Część wysyłanych pakietów inicjowana jest bezpośrednio przez protokoły pracujące w warstwie sieci, co oznacza, że docierając do hosta docelowego są analizowane również tylko i wyłącznie do warstwy trzeciej. Będą to między innymi pro-

tokoły ARP, RARP, DHCP oraz ICMP. Wykorzystanie tego ostatniego najczęściej kojarzymy z komendą **ping** lub **tracert**.

Pozostaje pytanie jak zaobserwować procesy, o których mówimy? Niestety bez odpowiednich aplikacji najczęściej będziemy mogli zauważyć jedynie wyniki ich działań. W przypadku systemu Windows osiągniemy to przy użyciu komendy **netstat**. Wywołując ją z parametrami **-s**, **-a** i **-r**, zobaczymy odpowiednio: statystyki dotyczące odbieranych i wysyłanych pakietów, informacje na temat połączeń i związanych z nimi numerów portów oraz trasy routingu (Rys. 2).

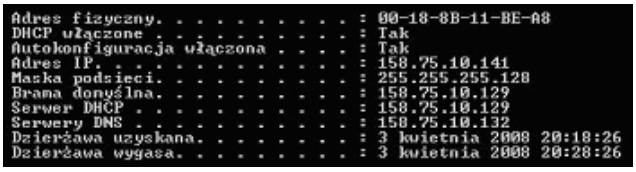
Rys.2 Wykorzystanie komendy **netstat –a**.

Warto zatem zaopatrzyć się w aplikację, która w znaczący sposób rozszerzy te możliwości. Wykorzystamy tutaj program Wireshark, który dostępny jest zarówno dla systemów Windows jak i Linux. Programy tego typu (tzw. snifery) przestawiają działanie systemu w taki sposób, aby możliwym było pełne analizowanie wszystkich danych trafiających do naszego interfejsu. Co więcej program ten pozwala również na interpretację otrzymywanych danych na tle modelu OSI, co znacznie ułatwia ich zrozumienie.

Prześledźmy teraz pełen zestaw danych, które są odbierane i przesyłane przez naszego hosta. Dla ułatwienia załóżmy, że włączamy komputer i chcemy sprawdzić jaka w dniu jutrzejszym będzie pogoda na naszym ulubionym portalu ICM (meteo.icm.edu.pl). Najczęściej wybieraną opcją dotyczącą konfiguracji interfejsu sieciowego jest konfiguracja automatyczna, co w większości przypadków oznacza, iż włączamy protokół DHCP (*ang. Dynamic Host Configuration Protocol*). W momencie uruchamiania systemu interfejs nie posiada przydzielonego adresu IP, a zatem jedyną pewną informacją jest adres MAC zapisany na karcie sieciowej. W związku z powyższym system rozpoczyna pracę od wysłania do serwera DHCP zapytania o konfigurację związaną ze swoim interfejsem. Zauważmy, iż system nie zna również adresu MAC i IP tego serwera, a zatem wysłany pakiet musi być zaadresowany w specjalny sposób. Okazuje się, że istnieją adresy rozgłoszeniowe, które są poprawnie weryfikowane niezależnie od faktycznego adresu interfejsu hosta. Adresy te składają się z samych jedynek, co oznacza, że w warstwie drugiej przybierają postać FF-FF-FF-FF-FF-FF, a w warstwie trzeciej 255.255.255.255, a nazywane są adresami rozgłoszeniowymi (*ang. broadcast)*. Ogólnie rzecz biorąc specyfika algorytmów weryfikujących adresy pozwala nam na dosyć precyzyjne określenie grupy

adresów hostów, na których weryfikacja będzie pomyślnie przeprowadzona, ustawiając wartość 1 dla bitów, które nie mają podlegać sprawdzeniu. Oznacza to, iż w przypadku, gdy chcemy, aby pakiet dotyczył tylko wszystkich hostów znajdujących się w podsieci 145.145.145.96 /27 (maska 255.255.255.224) musimy użyć adresu, który uzyskamy z po zmianie w adresie podsieci wszystkich bitów w polu hosta na wartość 1, czyli 145.145.145.127. Istnieją również adresy grupowe (*ang. multicast*), które przechodzą poprawną weryfikację tylko na hostach, które wspierają związane z nimi wybrane protokoły.

Wracając do opisywanej sytuacji oznacza to, iż nasz host musi w pierwszej kolejności wysłać zapytanie DHCP w postaci rozgłoszenia. W otrzymanej odpowiedzi od serwera DHCP (która jest pakietem typu *unicast*) zawarty jest nie tylko adres IP i maska potrzebne do konfiguracji naszego interfejsu, ale również informacje na temat adresu bramy oraz serwera DNS. Pamiętajmy, że w przypadku systemu Windows konfigurację interfejsu możemy zawsze zweryfikować przy użyciu komendy **ipconfig /all** (Rys.3).



Rys.3 Wykorzystanie komendy **ipconfig /all**

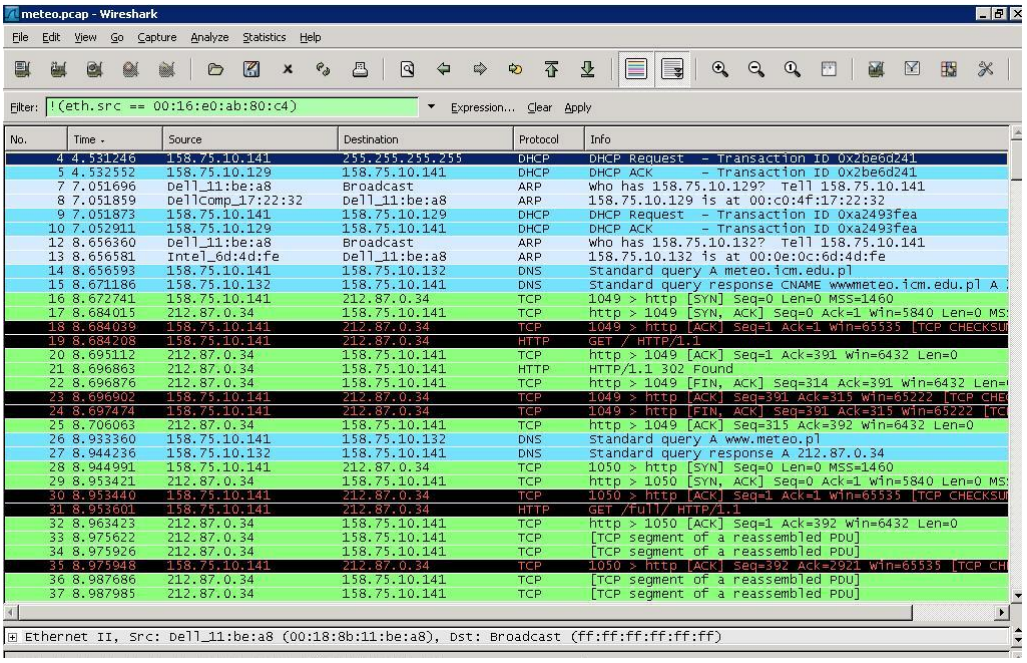
W celu sprawdzenia interesujących nas informacji uruchomiliśmy przeglądarkę internetową i wpisaliśmy w polu adresu nazwę serwisu meteo.icm.edu.pl. Protokół, którego domyślnie użyje nasza aplikacja to HTTP (*ang. Hypertext Transfer Protocol*). Niestety użyliśmy nazwy, a nie adresu IP, a zatem na wstępie nasz system musi wysłać zapytanie do serwera świadczącego usługę DNS (*ang. Domain Name System*) polegającą na interpretowaniu nazw jako adresów IP. Zapytanie to podlega również procesowi enkapsulacji, a zatem zanim to nastąpi musimy poznać również adres MAC tego serwera. W tym celu system wykorzystuje protokół ARP, generując zapytanie o adres MAC urządzenia, podając jednocześnie jego adres IP. Tutaj pojawia się problem, gdyż pakiet generowany jest przez warstwę trzecią, co oznacza, iż spodziewamy się odpowiedzi od hosta docelowego generowanej przez tą samą warstwę. Pamiętamy jednak, że aby dane dotarły do warstwy sieci na hoście docelowym, wcześniej ramka musi zostać zweryfikowana na poziomie adresu MAC (warstwa 2), a jego właśnie nie znamy. W tym miejscu jest, zatem czas na użycie adresu rozgłoszeniowego, co oznacza, iż każdy host w naszej sieci zweryfikuje tą ramkę w prawidłowy sposób na poziomie warstwy 2, a tylko serwer DNS zaakceptuje tak otrzymany pakiet na poziomie adresu IP i odpowie na niego zwrotnym pakietem ARP. Ten ostatni nie zawiera już adresów rozgłoszeniowych, a zatem będzie on poprawnie zweryfikowany tylko przez naszego hosta. Znając adres MAC i IP serwera DNS, nasz system wysyła do niego pakiet z prośbą o podanie adresu IP odpowiadającego serwerowi, który jest dostępny pod nazwą „meteo.icm.edu.pl". Po otrzymaniu informacji, że adres ten to 212.87.0.34, nasz system zauważy, iż adres

ten znajduje się poza naszą siecią, a zatem w generowanym pakiecie docelowy adres IP będzie miał wartość 212.87.0.34, natomiast docelowy adres MAC będzie odpowiadał adresowi fizycznemu interfejsowi bramy. W naszym przypadku warto zauważyć, iż rolę serwera DHCP oraz bramy pełni to samo urządzenie (Rys.3), którego adres MAC nasz host już zdążył poznać. W przypadku gdyby system nie posiadałby tych informacji, wygenerowałby kolejne rozgłoszenie będące zapytaniem ARP skierowanym tym razem do bramy. Ostatecznie po uzyskaniu niezbędnych informacji możemy wreszcie wysłać zapytanie do serwera WWW. Zanim jednak nastąpi proces przesyłania danych, należy zauważyć, że będziemy używać protokołu HTTP (warstwa 7), który jest wspierany przez protokół TCP (*ang. Transmission Control Protocol*) zapewniający kontrolę przepływu danych (warstwa 4). Na wstępie nawiązuje on połączenie wykorzystując system trójstopniowego uzgodnienia. Oznacza to, iż nasz host wysyła pakiet TCP do serwera WWW informując go, iż będzie chciał nawiązać połączenie, ten zaś nawiązując je odpowiada zwrotnie analogicznym pakietem. Pełne uzgodnienie nastąpi dopiero po wysłaniu zwrotnego pakietu TCP z hosta do serwera, który potwierdzi tą transmisję. Tak nawiązane połączenie jest oczywiście wirtualne, co oznacza, że jest ono tylko logicznym uzgodnieniem połączenia pomiędzy dwoma hostami i nie skutkuje ono określeniem jednoznacznej drogi pakietu określonej przez urządzenia sieciowe. Protokół TCP zapewnia transmisję wykorzystując potwierdzenia, co oznacza, że host źródłowy wysyła pierwszą porcję danych, a następnie przerywa nadawanie i czeka na potwierdzenie od hosta docelowego. Ten zaś po odebraniu danych wysyła zwrotnie informację, prosząc o kolejne dane. Bardzo istotną rolę odgrywają tutaj numery portów, synchronizacji (parametr SYN) i potwierdzeń (parametr ACK), które pozwalają zorientować się odbiorcy i nadawcy jakiej transmisji dotyczy potwierdzenie oraz, które dane mają zostać wysłane w kolejnych pakietach. W przypadku, gdy przesyłane dane zostały uszkodzone podczas transmisji, lub nie dotarły w całości do hosta docelowego, wyśle on pakiet TCP do hosta źródłowego, określając za pomocą numeru ACK numer oktetu od którego transmisja ma zostać wznowiona. Pozostaje jeszcze pytanie, w jak dużych porcjach chcemy wysyłać dane, po których mamy oczekiwać potwierdzenia. Wielkość ta określana jako okno (parametr WIN) uzgadniana jest podczas nawiązywania połączenia i może być modyfikowana podczas transmisji. Zwróćmy uwagę, iż w przypadku, gdy wartość okna zostanie ustawiona na większą niż, wartość parametru MTU, to host wysyłający dane wyśle je w kilku pakietach. Pamiętajmy jednak, że mówimy o warstwie czwartej, a parametr MTU określa wielkość całego pakietu, w którym oprócz danych zawarte są jeszcze inne pola warstwy trzeciej (między innymi adresy IP) oraz warstwy czwartej (między innymi numery portów). Oznacza to, iż każdy host przygotowując dane do wysłania musi wiedzieć jaką maksymalną wielkość (określoną w bajtach) może mieć pojedynczy segment. Wielkość ta określana jako MSS (*ang. Maximum Segment Size*) również zostaje uzgodniona pomiędzy dwoma hostami nawiązującymi połączenie. Nie chcąc doprowadzać podczas transmisji do dodatkowej fragmentacji danych wartość ta powinna być mniejsza niż parametr MTU (np. dla MTU=1500 typowo MSS=1460). W związku z powyższym, dla okna o wartości WIN=2960 wysłane zostaną dwa pakiety, po których host wysyłający dane będzie spodziewał się kolejnego potwierdzenia.

W celu wykorzystania maksymalnego pasma bardzo często okno przyjmuje wartość maksymalną czyli 65535, co oznacza, że host źródłowy domyślnie będzie spodziewał się potwierdzenia dopiero po wysłaniu 45 pakietów. Okazuje się jednak, iż w celu obliczenia parametru RTT (*ang. Round-Trip Time*) czyli czasu związanego z szybkością transmisji danych, nasz host może być zmuszony do cyklicznego wysyłania pakietów TCP potwierdzających dostarczenie kolejnych ramek. W przypadku, gdy serwer nie będzie miał już żadnych danych do przesłania, wyśle do hosta docelowego pakiet TCP z parametrem FIN, informując tym samym o zamiarze zakończenia połączenia. Połączenie jest dwustronne, zatem rozłączenie nastąpi dopiero po otrzymaniu potwierdzenia od hosta docelowego. Ta sekwencja spowoduje zamknięcie połączenia na serwerze. W celu całkowitego zamknięcia połączenia host docelowy rozpocznie analogiczną sekwencję jak serwer.

W opisany powyżej sposób host i serwer po uzgodnieniu połączenia, stopniowo przesyłają między sobą dane, które następnie zapisywane w plikach mogą być np. interpretowane przez przeglądarkę internetową, co ostatecznie użytkownik widzi na ekranie komputera. W przypadku gdy użytkownik zapragnie uzyskać więcej informacji klikając na kolejnym odnośniku zawartym na stronie WWW, proces rozpoczyna się na nowo od uzgodnienia kolejnego połączenia.

Zamieszczony poniżej Rys. 4 pokazuje, iż opisane tutaj pakiety można bez trudu zaobserwować i analizować uruchamiając wcześniej program Wireshark, co też gorąco polecam czytelnikowi.



Rys.4 Analiza pakietów za pomocą programu Wireshark

Zaprezentowany sposób przesyłania danych jest dosyć skomplikowany i przedstawiony w postaci suchej teorii na pewno nie będzie dobrze zrozumiany. Użycie programów typu Wireshark jako narzędzia dydaktycznego pozwala na rzeczywistą analizę sposobu działania poszczególnych aplikacji i związanych z nimi protokołów sieciowych. Co więcej pełna analiza przesyłanych danych na tle mode-

lu OSI umożliwia nie tylko badanie zależności między współdziałającymi procesami, ale również wskazuje jak w praktyce model ten ułatwia zrozumienie bardzo złożonych procesów. Dzięki takim programom użytkownik na własne oczy może przekonać, się ile złożonych czynności jest realizowanych podczas zwykłego oglądania stron internetowych, czy też wysyłania wiadomości. Co ważniejsze lepsze zrozumienie procesów realizowanych przez nasz system podczas przesyłania danych, pozwala również na trafniejszą ocenę przyczyn ewentualnych problemów pojawiających się przy korzystaniu z sieci komputerowych. Należy również zaznaczyć, iż opisywane oprogramowanie objęte jest licencją GNU (*ang. General Public License*), co oznacza, iż proponowane testy może przeprowadzić każdy posiadacz komputera podłączonego do sieci komputerowej, nie ponosząc przy tym żadnych dodatkowych kosztów.

4. Kilka słów na temat monitorowania danych

Pozostaje jednakże pytanie, czy możliwym jest śledzenie samych danych związanych z konkretną komunikacją, bez oglądania całych datagramów. Okazuje się, iż w opisywanej aplikacji niezastąpionym staje się w tym przypadku opcja umożliwiająca śledzenie strumienia danych (Follow TCP Stream). Pozwala ona na wybranie ze wszystkich przechwyconych ramek tylko tych, które dotyczą interesującej nas konwersacji, a następnie złożenie ich w jeden plik tekstowy. Jak widać użyta aplikacja jest bardzo uniwersalna i pozwala na praktyczne poznanie sposobu działania różnych protokołów, a nawet przeglądanie przesyłanych przez nie treści. Dzięki temu możliwym staje się również wyciąganie wniosków na temat ich bezpieczeństwa, co ze względu na ograniczoną formę tej pracy pozostawiam już czytelnikowi do samodzielnej oceny. Na zakończenie pragnę tylko dodać, iż przesyłając dane w postaci e-mail'a, czy też wpisując je w formularzu na stronie WWW, musimy mieć świadomość, iż nie mamy żadnej kontroli nad tym czy ktoś niepowołany nie uzyska ich kopii. Oznacza to, iż w przypadku, używania protokołów przesyłających dane tzw. otwartym tekstem, musimy mieć świadomość, iż każdy, kto wejdzie w ich posiadanie może je bez trudu odczytać. Nie mając kontroli nad naszymi danymi w sieci zewnętrznej pozostaje nam jedynie ich kryptowanie. W celu zrozumienia różnicy w postaci tak przesyłanych danych proponuję czytelnikowi porównać dane strumieni przesyłanych np. za pomocą protokołów HTTP i HTTPS. Pamiętajmy jednak, iż nigdy nie jesteśmy w pełni bezpieczni, gdyż wszystko co zostało zaszyfrowane można odszyfrować. Problem polega tylko na możliwościach obliczeniowych komputerów, skończonym czasie obliczeń oraz odpowiednich algorytmach, a zatem zawsze przede wszystkim należy zachować zdrowy rozsądek i dużą ostrożność.

Literatura

1. M. Piwiński, Uczniowie i komputery w sieci, Komputer w Szkole, nr 5, (2003), s 38-45
2. dokument RFC 793 dostępny pod adresem http://tools.ietf.org/html/rfc793
3. http://en.wikipedia.org/wiki/Transmission_Control_Protocol
4. www.wireshark.org