

UNIwersytet WarMińsko-Mazurski
W Olsztynie

Wydział Matematyki i Informatyki

Kierunek: Informatyka

Górski Adam.

Wiarygodność transmisji bitowej w zbiorze hostów
wykorzystujące media sieciowe do publikacji danych.

Praca inżynierska została
wykonana w katedrze
Elektrotechniki, Energetyki,
Elektroniki i Automatyki
pod kierunkiem
dr inż. Zenon Syroka.

Olsztyn 2022

UNIVERSITY OF WARMIA AND MAZURY
IN OLSZTYN
FACULTY OF MATEMATICS AND INFORMATICS

Direction: Informatic

Górski Adam.

Reliability of the bit transmission of hosts using network
media to publish data.

Engineer Thesis written in
Department of Electrotechnics,
Power Engineering, Electronics and
Automation under supervision of
dr inż. Zenon Syroka.

Olsztyn 2022

Strona 1 z 68

Spis treści	
Streszczenie.....	3
Summary.....	5
1 Analiza i przechwytywanie ruchu sieciowego.....	6
1.1 Protokoły IPv4 i IPv6.....	6
1.1.1 IPv4.....	9
1.1.2 IPv6.....	11
1.2 Opis najważniejszych protokołów.....	12
1.3 Bierne przechwytywanie ruchu sieciowego.....	17
1.4 Aktywne przechwytywanie ruchu sieciowego.....	24
1.5 Podsumowanie.....	33
2 Szyfrowanie i kodowanie danych.....	35
2.1 Starożytne początki zabezpieczania informacji.....	36
2.2 Podstawowe szyfry.....	38
2.3 Problem szyfru z kluczem jednorazowym. Jak go rozwiązać?	49
2.4 Algorytmy kryptograficzne w cyfrowym świecie.....	51
2.5 Podsumowanie.....	55
3 Program do zbierania informacji o trasie pakietów.....	56
4 Opis działania programu.....	60
Podsumowanie.....	63
Summary.....	63
Spis Rysunków.....	64
Bibliografia.....	66

Table of Contents

Summary in Polish.	3
Summary in English.....	5
1 Network Analysis and Network Traffic Capture.	6
1.1 Protocol IPv4 and IPv6.....	6
1.1.1 IPv4.....	9
1.1.2 IPv6.....	11
1.2 Description The Most Important Protocol.....	12
1.3 Passive Network Traffic Capture	17
1.4 Active Network Traffic Capture.....	24
1.5 Summary.....	33
2 Encryption and Encoding Data.	35
2.1 Ancient Genesis Securing Data.	36
2.2 Basic Ciphers.....	38
2.3 Cipher Issue One-Time Pad. How To Resolve It?	49
2.4 Cryptographic Algorithms In Digital World.	51
2.5 Summary.....	55
3 Program To Collect Informations About Network Traffic.	56
4 Description Program.....	60
Summary in Polish	63
Summary in English.....	63
Table of Drawings	64
Bibliography.....	66

Streszczenie.

Poniższa praca składa się z czterech rozdziałów. Dwa pierwsze rozdziały poświęcone są teoretycznym rozważaniom dotyczącym elementów które wpływają na bezpieczną transmisję bitową. Natomiast rozdział trzeci i czwarty opisuje program, który został napisany bazując na zdobytej wiedzy podczas pisania pracy.

W rozdziale pierwszym znajdziemy informację o najważniejszych protokołach sieciowych które uznałem za słuszne oraz ich dokładną analizę. Dodatkowo rozdział pierwszy zawiera drugi ważny aspekt bezpiecznej transmisji, to jego analiza. Przedstawiłem tam sposoby analizy ruchu sieciowego w sposób bierny jak i aktywny dzięki czemu można mieć pełen wgląd w ruch sieciowy.

Rozdział drugi to również ważny aspekt ruchu sieciowego dotyczy on szyfrowania i kodowania danych. Wewnątrz rozdziału znajdziemy trochę historii dotyczącej starożytnych sposobów zabezpieczenia informacji. Opisałem tam historyczne narzędzia, które w ciekawy sposób potrafią zmienić sens wiadomości tak aby można było bezpiecznie i niezauważalnie dostarczyć do miejsca docelowego. Następnie opowiedziałem o podstawowych szyfrach które nadal mogą pełnić ważną rolę w zabezpieczaniu informacji. Opisałem tam szyfr cezara jak i wiele innych, które wpłynęły na rozwój kryptografii. W kolejnym podrozdziale rozważyłem sposób rozwiązania problemu z szyfru z kluczem jednorazowym, gdzie przedstawiłem swój punkt widzenia na ten temat. W ostatnim zaś podrozdziale przeszedłem do ważnych aspektów kryptografii w dzisiejszym świecie opisując najważniejsze algorytmy kryptograficzne. Całość zwięźliłem podsumowaniem.

Rozdział trzeci to przedstawienie mojego programu, który wykorzystuje zebraną wiedzę podczas pisania pracy. Opisałem tam kod najważniejszych funkcji tak aby w jasny sposób przedstawić ważne aspekty mojego programu.

Rozdział czwarty to opis działania programu oraz jego funkcji.

Na końcu pracy przedstawiłem podsumowanie całej mojej pracy oraz co zostało osiągnięte.

Summary.

My Engineering work include four chapters. The first two chapters contains theoretical speculation about safety network traffic. However chapters third and four contains program descriptions. Which is based on knowledge from chapters first and second.

The first chapters contains information about the most important protocol in the Internet and their analysis. We can to find into the chapters analysis network traffic capture passive, active and describe how do that.

The second chapters describes one of the most important aspects network traffic capture Encoding and encryption data. I show it history encryption data and ancient encryption tool. Also describes Cezar cipher and show how it work. Additionally describe into the chapters modern algorithm nowadays such as AES RSA and DES.

The third chapter present my application. This application is based of the knowledge the chapters first. I describe into the most important part of code.

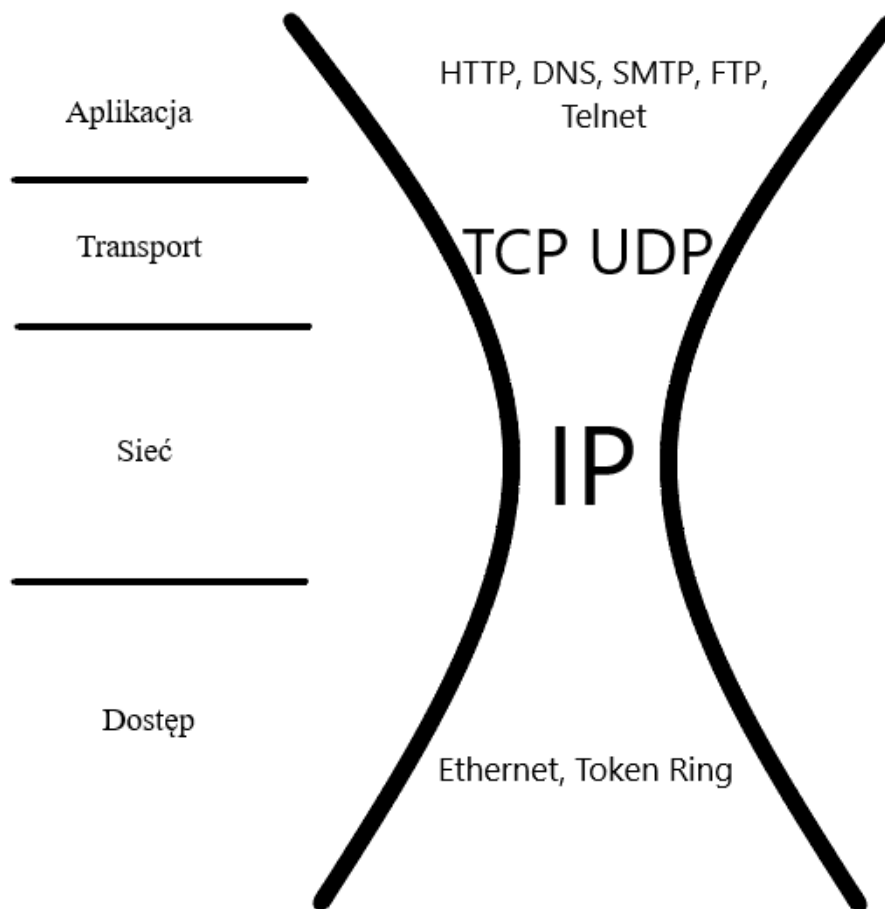
The fourth chapter is describe application and her functions.

1 Analiza i przechwytywanie ruchu sieciowego.

Gdy się uczymy czytać najpierw poznajemy poszczególne litery. Następnie uczymy się wymawiać potem składać sylaby, a na końcu czytać całe wyrazy. Znajomość najprostszych elementów tak zwanych fundamentów pozwala osadzać nabywaną wiedzę na dobrym podłożu. Ucząc się każdej nowej rzeczy zawsze, ale to zawsze zaczynamy od elementów najprostszych czasami od historii powstania. Często również poznajemy historyczne wersje danej rzeczy. Jeśli chcemy poznać budowę silnika najpierw uczymy się najprostszych schematów obrazujące jego działanie następnie poznajemy bardziej skomplikowane elementy. Dlatego rozdział dotyczący przechwytywania ruchu sieciowego rozpoczniemy od podstaw. Od elementów budowy następnie analizując te elementy przechodząc do zasady działania kończąc na wadach, które pozwalają wykorzystać słabości w celu ataku na poszczególne cele sieciowe. Na końcu postaram się wykorzystać zebraną wiedzę, która jest niezbędna dla zapewnienia bezpieczeństwa sieci. Analiza ruchu sieciowego bez znajomości podstaw działania jest jak oglądanie elementu, który nie wiemy do czego ma służyć. Jeśli posiadamy w ręku broń, a nie wiemy, jak jej użyć to należy się liczyć, że możemy zrobić sobie krzywdę podczas jej używania. Dlatego najważniejsze to poznać podstawy dopiero uczyć się zaawansowanych rzeczy. Dzięki ich poznaniu możemy znajdować luki w systemach i skutecznie je eliminować po to, aby użytkowanie Internetu było bezpieczne. Cały poniższy rozdział skupia w sobie wszystkie najważniejsze elementy jakie powinien znać informatyk podczas analizy i przechwytywania ruchu sieciowego.

1.1 Protokoły IPv4 i IPv6.

Znajomość protokołu IP w analizie ruchu sieciowego jest przydatna jak dodawanie w codziennym życiu bez znajomości fundamentów chociażbyśmy mieli najlepsze narzędzia budowlane to nic nie zdołamy osiągnąć. Dlatego też swoją pracę inżynierską zaczynam od analizy fundamentów w dziedzinie wiarygodności transmisji bitowej. W zasobach internetowych istnieje wiele protokołów i rozwiązań dostępnych które możemy bez problemu



Rysunek 1.1. Stos protokołów TCP/IP w formie klepsydry.

wykorzystać. Wszystkie te rozwiązania zostały ujęte w różne stosy, do których np. należy wynaleziony w latach 70 w Stanach Zjednoczonych Ameryki Północnej czterowarstwowy model Departamentu Obrony USA czy późniejszy OSI. W poniższym rysunku można dostrzec, że wszystkie warstwy górne i dolne zawierają mnóstwo protokołów i rozwiązań, które naprawdę ciężko spamiętać, a jest ich o wiele więcej. Ktoś kto zna budowę tych stosów wie, że protokół IP leży w warstwie drugiej modelu TCP/IP bądź trzeciej OSI. Więc może wydawać się dziwne, dlaczego zaczynam od protokołu warstwy trzeciej, a nie klasycznie od warstwy pierwszej potem druga itd. Otóż powód jest prosty protokół IP czy to w wersji czwartej albo szóstej nie mają konkurencji są niezastąpione. Zobrazować to może rysunek 1.1 który przedstawia nam stos, który nie bez przyczyny porównałem do klepsydry. Dlaczego protokół IP w wersji czwartej lub nowszy w szóstej jest tak ważny, że w mojej pracy znajduje się na pierwszym miejscu? Otóż całą tą sytuację możemy porównać do codziennego życia. Internet

nie bez przyczyny to taki wirtualny świat. Nadawca listu to można powiedzieć życiowa warstwa aplikacji, która chce wymienić dane z inną warstwą aplikacji. Mając naszą wiadomość wybieramy sposób dostarczenia tych informacji np. list priorytetowy, list zwykły paczka, przesyłka itp. TCP może odwzorowywać list priorytetowy którego odbiór zostanie potwierdzony, UDP to list zwykły tutaj firmy nie interesuje jego utrata to my jako warstwa aplikacji musimy zapewnić, że dotrze np. zadzwonić zapytać się czy jest w razie potrzeby ponowić wysyłanie. Przejdę teraz do warstwy Dostępu Do sieci tutaj odwzorowanie jest dosyć prosty sposób przenoszenia np. samochód, samolot, rower, prom itp. Znowu zauważamy dużo możliwości i różnych opcji wyboru. Zatem jak pokazać IP w wersji życiowej otóż to jest adres naszego domu, który bezpośrednio identyfikuje miejsce, w które ma trafić przesyłka tak jak adres IP w sieci. Jak można zauważyć istnieje wiele rzeczy w poszczególnych warstwach oczywiście ktoś może powiedzieć, że w warstwie sieciowej znajdują się inne protokoły tak, lecz nie pełnią tak ważnej roli jak protokół IP. Protokół IP jest to element, który spaja sieć w jedną całość gdybyśmy mieli ją kroić na pół jak owoc to z jednej strony sieci byłoby I z drugiej P. IP tak jak adres domu spaja sieć, albowiem warstwę aplikacji nie interesuje jakim sposobem, którędy dostarczymy wiadomość tylko ona daje dane zawartość mówi jakim typem transmisji chce ją dostarczyć z kim lubi współpracować i więcej ją nie interesuje tak samo jak firma która przewozi przesyłkę nie interesuje ją co jest w środku oraz co tam znajdzie. To właśnie adres spaja te dwie rzeczy powoduje, że mimo wielu technologii dostępnych pomiędzy znajduje się coś co pozwala na połączenie tego w jedną logiczną działającą całość tak aby mogło to działać. Dlatego protokół IP jest tak ważny, bo pomimo wielu technologii sposobów dostawy, aplikacji jest coś co pozwala spoić pozwala kontrolować to wszystko, aby działało nawet przy tak zróżnicowanych technologiach, które mamy do wyboru. Nawet jeśli nadajemy wiadomość z Polski do USA i ona podróżuje różnymi drogami to wszystko wie, że musi dostać się do określonego miejsca i tam będą wiedzieli co z tym mają zrobić. Dlatego protokół IP jest ważny on po prostu nadzoruje obie strony dzięki czemu sieć mimo tak skomplikowana i rozbudowana może skutecznie działać.

1.11 IPv4

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
00	1 wersja				2 IHL				3 ToS								4 długość całkowita															
32	5 identyfikator																6 flagi			7 przesunięcie fragmentu												
64	8 TTL								9 protokół								10 suma kontrolna nagłówka															
96	11 adres źródłowy IP																															
128	12 adres docelowy IP																															

Rysunek 1.2. Budowa nagłówka IPv4.

Nagłówek IPv4 składa się z 12 części:

- 1) Oznacza wersję protokołu w tym przypadku będzie to 0100, czyli 4.
- 2) Długość nagłówka- minimalna wartość to 5 a maksymalna to 15. Liczbę tę mnożymy razy 4 i mamy rozmiar naszego nagłówka, czyli wiemy, ile miejsca na kopercie zajmuje adres
- 3) Typ Usługi: Bity od 08 do 13 odnoszą się bezpośrednio poziomowi i rodzajowi świadczonej usługi, bity od 8 do 10 mówią jak istotny jest to pakiet (0 mało istotny-7 arcyważny), bit 11 oznacza potrzebę małego opóźnienia, bit 12 potrzebna duża przepustowość, bit 13 oznacza żądanie dużej niezawodności
- 4) Rozmiar wraz z danymi minimalny 576 maksymalny 65535
- 5) Oznaczenie Pakietu te same części mają te same oznaczenia co w jednoznaczny sposób je Identyfikuje i wiadomo do czego należą.
- 6) Bit 16 nieużywany, 17- DF nie dziel, 18 istnieje więcej części, ta flaga jest wszędzie ustawiona oprócz ostatniej części, gdzie po niej już nie ma więcej.
- 7) Oznacza w którym miejscu znajduje się dana część
- 8) Czas życia- wartość ta jest zmniejszana przez każdy ruter, a gdy dojdzie do zera pakiet jest porzucany
- 9) Protokół wyższej warstwy TCP to 6 a UDP to 17
- 10) Suma
- 11) Skąd wychodzi dany pakiet (miejsce w sieci)
- 12) Dokąd idzie dany pakiet (miejsce w sieci)

Adresowanie:

Adres w IPv4 składa się z czterech oktetów razem 32 bity. IPv4 zostało podzielone na 5 klas. Liczba wszystkich możliwych adresów w protokole IPv4 to 2^{32} .

Klasa	Zakres	Maska
A	1.0.0.0-126.0.0.0	255.0.0.0
B	128.1.0.0-191.254.0.0	255.255.0.0
C	192.0.1.0-223.255.254.0	255.255.255.0
D	224.0.0.0-239.255.255.254	-
E	240.0.0.0-255.255.255.255	-

*adresy od 127.0.0.1 do 127.255.255.254 są nierutowane używane do pętli zwrotnej

Rysunek 1.3. Podział adresów IPv4.

Do adresacji urządzeń sieciowych używa się zakresów klasy A, B i C klasy D i E należą do grupy adresów specjalnych, gdzie D to odpowiada za adresacje typu Multicast co pozwala na wysyłanie pakietów do wielu urządzeń, a E to adresy zarezerwowane. Mimo istnienia klas A, B oraz C w dzisiejszych czasach używa się pojęcia „bezklasowości” co sprawia, że adresowanie staje się bardziej elastyczne, ponieważ maska zmienia się w zależności od liczby hostów, a nie zależy od klasy adresu. Klasy D oraz E nie mają maski. Wyróżniamy również trzy zakresy które są adresami prywatnymi do użytku w sieciach typu LAN:

10.0.0.0 - 10.255.255.255

172.16.0.0 - 172.31.255.255

192.168.0.0 - 192.168.255.255

Najważniejsze zagadnienia:

Adres sieci- IP które w sposób unikatowy rozpoznaje daną sieć

Maska- jak wskazuje słowo służy do ukrywania czegoś. Maska pozwala znaleźć wiele istotnych informacji takie jak:

- dla podanego adresu IP adres sieci, adres rozgłoszeniowy,
- sprawdzić czy dany adres odpowiada danej sieci,
- odczytać liczbę komputerów w sieci,
- dokonać podziału na podsieci.

Adres rozgłoszeniowy- odpowiada za komunikacje ze wszystkimi użytkownikami w sieci, również możemy spotkać:

Adres hosta- adres każdego urządzenia który ma przydzielone IP,

Adres bramy- urządzenie, które odpowiada za komunikacje z następną siecią (domeną rozgłoszeniową),

Adres podsieci- IP które w unikatowo opisuje konkretną podsieć.

1.1.2 IPv6

Bity	0-3	4-7	8-11	12-15	16-19	20-23	24-27	28-31
0	1 Wersja	2 Usługi zróżnicowane		3 Etykieta przepływu				
32	4 Długość danych				5 Następny nagłówek		6 Limit przeskoków	
64	7 Adres źródłowy 128 bitów							
96								
128								
160								
192	8 Adres docelowy 128 bitów							
224								
256								
288								

Rysunek 1.4 Nagłówek protokołu IPv6.

Nagłówek protokołu IPv6 składa się z 8 części:

- 1) Liczba binarna oznaczająca wersję protokołu
- 2) Odpowiada za jakość dostarczonego pakietu (naszego listu) oraz informuje o zatorach.
- 3) Służy do oznaczania pakietów które mają wspólne wymagania.
- 4) Wielkość danych przesyłanych przez protokół IPv6 maksymalnie 65 535.
- 5) Określa typ następnego nagłówka
- 6) Czas życia- wartość ta jest zmniejszana przez każdy ruter, a gdy dojdzie do zera pakiet jest porzucany
- 7) Skąd wychodzi dany pakiet (miejsce w sieci)
- 8) Dokąd idzie dany pakiet (miejsce w sieci)

Adres IPv6 składa się z 128 bitów co pozwala na zaadresowanie 2^{128} urządzeń w sieci. Jest podzielony na 8 równych części każda z nich ma długość 16 bajtów, a do jego notacji wykorzystuje się system szesnastkowy np. fe80::543b:5053:ca2c:dd85::12. Separatorem jest dwukropek oraz możemy pomijać zera dzięki czemu zapis staje się krótszy np.

2001:0db8:0000:0000:0000:0000:0345:0090 = 2001:0db8:0:0:0:0:0345:90 =

2001:0db8:0:0:0:0:0345:90 = 2001:0db8::0345:90 = 2001:0db8::345:90. Wszystkie te adresy mimo różnych zapisów przedstawiają ten sam adres sieciowy. Kilka ważnych adresów IPv6:

fe80::/10 i fc00::/7 - pula adresów prywatnych

::1/128- pętla zwrotna

2001:0db8::/32 i 3fff:ffff::/32 - adresy służące do rozważań teoretycznych oraz dokumentacji,

ff00::/8- komunikacja Multicast.

O protokole IP można byłoby napisać jeszcze wiele istotnych i ważnych informacji, lecz uważam, że najważniejsze są podstawy i fundamenty. Wiele więcej informacji znajdziemy w książkach, lecz dobrze wytłumaczone podstawy w prosty sposób to rzadkość. W książkach czy publikacjach internetowych mamy do czynienia z przytaczaną definicją, która przechodzi tylko permutacje słów. Dlatego też w swojej pracy piszę prosto i konkretnie co jak wiadomo jest najtrudniejsze do realizacji.

1.2 Opis najważniejszych protokołów.

Warstwa	Protokoły
Aplikacji	DNS, DHCP, HTTPs
Transportu	TCP, UDP
Sieci	IP, ARP/NB, ICMP/v6
Dostępu	Ethernet II

Rysunek 1.5 Najważniejsze protokoły z podziałem na poszczególne warstwy.

DNS- Protokół, którego komunikacja przebiega na porcie 53. Służy do odwzorowywania\ tłumaczenia adresu IP na nazwy przystępne dla człowieka. Każdemu z nas łatwiej jest zapamiętać, że jabłko ma kolor np. czerwony niż pamiętać 255-0-0 (RGB) tak samo w sieci łatwiej zapamiętać nam słowo niż adres IP 82.65.1.0. DNS powstał, ponieważ poprzedni sposób z rozwiązywaniem nazw za pomocą pliku host stawał się nieefektywny. Plik ten wraz z rozwojem rósł do olbrzymich rozmiarów co powodowało, że czas jego przeszukiwania powodował problemy z wydajnością. Mimo to plik host nadal został, a komputery nadal go przeszukują, gdy potrzebujemy znów przetłumaczyć i poznać jaki adres reprezentuje dana nazwa mnemoniczna. DNS jest protokołem szczególnym biorąc pod uwagę aspekty bezpieczeństwa, ponieważ wystarczy podczas tłumaczenia zmienić interesujący nas adres, a można sprawić, że w prosty sposób wyłudźmy dane podszywając się pod znaną stronę nawet najbardziej zabezpieczoną. Nawet jeśli nie można złamać systemu, który działa zawsze tak samo to człowiek to nie system zawsze może mieć gorszy dzień może brak czasu i wtedy, gdy jesteśmy słabsi podejmujemy decyzje, które potrafią być nierozważne co może wykorzystywać przestępca komputerowy w swoim działaniu.

DHCP- Protokół, który działa wykorzystując dwa porty 67 i 68. Pakiety wychodzą z portu 68 klienta, gdzie trafiają na port docelowy 67 w serwerze. Odpowiada on za automatyczne przydzielanie adresów w sieci komputerowej. Komputer podłączony do sieci zgłasza się do serwera DHCP wykorzystując port 68 z takim o to komunikatem DHCP DISCOVER. Teraz następuje odpowiedź serwera DHCP w której oferuje adres klientowi komunikat DHCP OFFER. Tu może nasunąć się małe pytanie jak serwer może komunikować się z klientem, skoro on nie ma adresu? Cały zabieg wymiany jest dokonywany za pomocą rozgłaszania to znaczy, że informacje docierają do wszystkich w sieci, a tylko dany komputer je przyjmuje. Jest to wygodne, ale nie wszystko co jest wygodne i automatyczne w sieci potrafi być bezpieczne dlatego że każdy komputer może przechwycić te dane. Przechwycenie danych nie jest tak niebezpieczne co podanie własnych danych dzięki czemu komputer może nieświadomie należeć do innej sieci. Następnym krokiem jest odpowiedź klienta, czyli DHCP REQUEST w którym klient żąda określonych danych od serwera nadal wysyłany w formie ROZGŁASZANIA. Dopiero pakiet DHCP ACK z potwierdzeniem wysyłanym przez serwer jest wysyłane do danego komputera, czyli ten pakiet dopiero trafia do jednego hosta, który powinien mieć skonfigurowany interfejs sieciowy według swoich ustaleń.

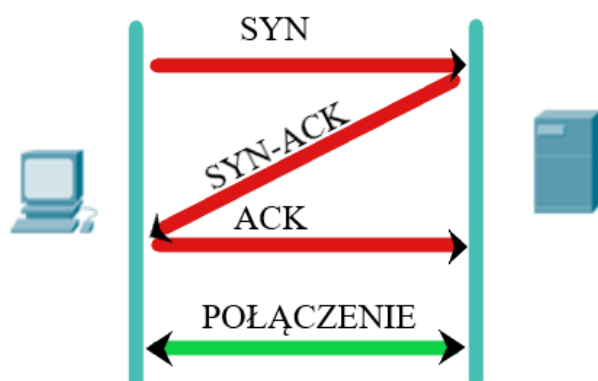
HTTP - Protokół, którego komunikacja odbywa się na porcie 80. Jest to najczęściej używany protokół w warstwie aplikacji. Służy do przesyłania hipertekstu odpowiada za poprawną komunikację pomiędzy klientem (przeglądarka), a serwerem WWW co w wyniku powoduje wyświetlenie strony internetowej. HTTP do swojej pracy używa protokołu połączeniowego TCP co oznacza, że wszystkie dane trafiają do odbiorcy. Protokół ten używa specjalnych kodów odpowiedzi, które w dużym uproszczeniu oznaczają:

- 1** kod informacyjny,
- 2** kod powodzenia,
- 3** kod przekierowania,
- 4** kod błędu po stronie klienta,
- 5** kod błędu serwera.

Rysunek 1.6. Kody odpowiedzi http.

HTTP, mimo że wydaje się być protokołem którym ciężko manipulować, lecz jak to zawsze bywa każdy element ma swoje wady i zdarzały się przypadki podmiany stron.

TCP- Protokół warstwy transportowej który odpowiada za przesłanie pakietów pomiędzy nadawcą i odbiorcą. Jest to protokół, który umożliwia komunikację bez straty informacji wiemy, że pakiety wszystkie dojdą do określonego celu. TCP wykorzystuje do swojej pracy



Rysunek 1.7. Uścisk dłoni TCP.

połączenie oparte na 3 prostych krokach tzw. three way handshake można to przetłumaczyć jako trzykrotny uścisk dłoni. Polega to na wymianie trzech komunikatów które są naszymi uściskami pomiędzy klientem, a serwerem. Protokół ten oprócz niezawodnego dostarczenia pakietów oferuje również bardziej mroczne wykorzystanie otóż elementy ustanawiania połączenia często są wykorzystywane do skanowania portów serwera. Gdy skaner wysyła pakiet SYN to oznacza, że chcemy połączyć się z danym portem, jeśli serwer odpowiada to oznacza, że istnieje usługa na tym porcie którą można wykorzystać np. do ataku. Zaletą jest to, że niewłaściwie zabezpieczony serwer nie zarejestruje tego, ponieważ połączenie nie zostało nawiązane więc komunikacji nie było co utrudnia administratorowi wykryć potencjalne niebezpieczeństwo, które może nastąpić. Oczywiście można wykorzystać całe połączenie TCP, lecz wtedy łatwo zwrócić uwagę na to, że dany adres chciał połączyć się z nami na wielu portach. Skanowanie z SYN można wykryć, lecz jest ono dużo trudniejsze i niedoświadczony administrator może jego nie zauważyć bądź zostawić tą lukę. Element ten też może zostać wykorzystany do ataku z odmową usługi wtedy, gdy atakujący wysyła pakiety SYN co powoduje, że serwer po odesłaniu SYN-ACK czeka na ACK, gdy wyślemy więcej pakietów SYN do serwera niż może serwer je obsłużyć nastąpi odmowa usług, czyli atak DDos.

UDP- Kolejny protokół warstwy Transportu. Pozwala dużo szybciej przesyłać dane, lecz nie gwarantuje ich dostarczenia. Z racji tego pozwala na wiele możliwości o tuż nie mamy pewności, że dany pakiet nie zostanie przechwycony i podmieniony co jest ogromną luką, wiedząc, że aplikacje np. DNS wykorzystuje do swej pracy UDP pozwala na przepiękny atak, po którym klient dostanie adres do naszej fałszywej strony www. Drugą możliwością jest wykorzystanie UDP jako skaner. Pomimo że UDP jest protokołem bezpołączeniowym to próba

połączenia z zamkniętym portem może spowodować wysłanie pakietu ICMP_PORT_UNREACH co pozwala dowiedzieć się które porty są nieaktywne więc reszta to ta która nas interesuje ze względu na możliwość ataku, ponieważ są to oczywiście otwarte porty.

IP- Tak jak napisałem, że jest to jeden z najważniejszych protokołów, ponieważ spaja sieć w jedną logiczną całość. Jest to protokół bezpołączeniowy co oznacza, że nie ma pewności, że pakiety trafią do miejsca docelowego. IP jest tak jak napisałem najważniejszym protokołem, a z racji tego jego nagłówek jest bardzo ważnym elementem analizy ruchu sieciowego, ponieważ dostarcza wielu cennych informacji.

ICMP- Protokół kontroli łącza. Informuje o zaistniałych problemach bądź błędach podczas transmisji. ICMP jest w budowie bardzo prosty co można zauważyć na poniższym rysunku:

1 Typ Komunikatu	2 Kod	3 Suma Kontrolna
------------------	-------	------------------

Rysunek 1.8 Budowa datagramu ICMP.

- 1) Zawiera informacje o błędzie.
- 2) Przenosi dodatkowe informacje.
- 3) Służy do sprawdzenia czy komunikat dotarł w nienaruszony sposób.

ICMP jest to protokół, który w przeciwieństwie do TCP nie gwarantuje, że dany pakiet dojdzie bądź nie zostanie zgubiony.

ARP- Służy do ustalania na podstawie danego adresu IP adresu MAC. Jest to ważny protokół w warstwie dostępu do łącza. Wykorzystuje się go podczas komunikacji, gdy jeden użytkownik chce wysłać pakiet do drugiego to, pomimo że zna adres IP musi jeszcze znać unikalny adres MAC, czyli tzw. pesel sieciowy który jest unikalny dla danego urządzenia w całej globalnej sieci. ARP można z powodzeniem wykorzystać do zmiany trasy pakietów zatruwając tablice ARP co pokaże w podrozdziale 1.4 Aktywne przechwytywanie ruchu sieciowego.

1 Typ sprzętu		2 Typ protokołu
3 Długość adresu sprzętowego	4 Długość adresu protokołu	5 Pole operacji
6 adres sprzętowy nadawcy		
7 adres IP nadawcy pakietu		
8 adres sprzętowy odbiorcy		
9 adres IP odbiorcy		

Rysunek 1.9. Budowa nagłówka ARP.

- 1) Rodzaj sprzętu w warstwie dostępu,
- 2) Rodzaj protokołu w warstwie sieciowej,
- 3) Długość adresu w warstwie dostępu,
- 4) Długość adresu w warstwie sieciowej,
- 5) Informacja o typie komunikatu (pytanie/odpowiedź),
- 6) MAC – nadawcy,
- 7) IP – nadawcy,
- 8) MAC – odbiorcy,
- 9) IP – odbiorcy.

Protokół ARP z pewnością ułatwia pracę, lecz jak każda rzecz, która przyspiesza i oszczędza pracę doprowadza do powstawania ruchu sieciowego który potrafi dostarczyć wielu cennych informacji oraz luk w zabezpieczeniach, dlatego tam, gdzie tylko możliwe należy wszystko robić ręcznie.

Ethernet II- Protokół warstwy dostępu do sieci. Jest to bardzo ważny protokół, ponieważ przygotowuje i dba o dobrą transmisję danych w sieci.

Format ramki Ethernet II

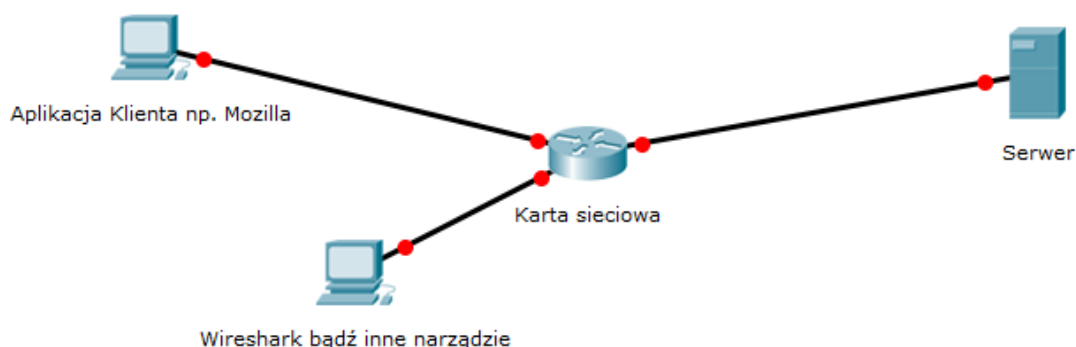
Preambuła	Adres docelowy	Adres źródłowy	Typ ramki	Dane ramki	CRC
-----------	----------------	----------------	-----------	------------	-----

Rysunek 1.10 Format ramki Ethernet II.

Początek tej ramki stanowi **preambuła**, która odpowiada za prawidłowe przygotowanie synchronizacji taktami zegara pomiędzy dwoma urządzeniami. Następnie musimy wiedzieć kto z kim się komunikuje tak aby wiedzieć, gdzie wysłać ramkę oraz komu wysłać odpowiedź i do tego służą adresy sprzętowe **MAC** docelowy i źródłowy. Gdy znamy drogę do którego hosta się udać to ramka Ethernet opisuje też co znajdziemy w środku np. kod 0x800 w polu Typ oznacza, że mamy do czynienia wewnątrz z protokołem IPv4. Dalej znajdują się dane, a na końcu CRC w celu sprawdzenia poprawności danych. Ramka Ethernet oprócz tego, że synchronizuje urządzenia to też ma określoną minimalny rozmiar 64 bajty co daje odpowiednio długi czas wysłania pakietu. Wszystko to nie jest bez znaczenia minimalny rozmiar jest potrzebny, aby w razie wystąpienia kolizji móc rozróżnić poprawne ramki od śmieci. Jest też inny powód, aby transmisja trwała odpowiednio długo. Wtedy, kiedy dojdzie do kolizji i zostanie wysłany sygnał zagłuszający musi on dotrzeć do nadawcy przed zakończeniem transmisji wtedy nadawca wie, że taka kolizja nastąpiła. Gdyby tak nie było nadawca mógłby po prostu nie zauważyć, że nastąpiła kolizja i dalej wysyłałby swoje dane co mogłoby powodować kolejne kolizje. Maksymalny rozmiar bez preambuły to 1518 bajtów, a z 1525.

1.3 Bierne przechwytywanie ruchu sieciowego.

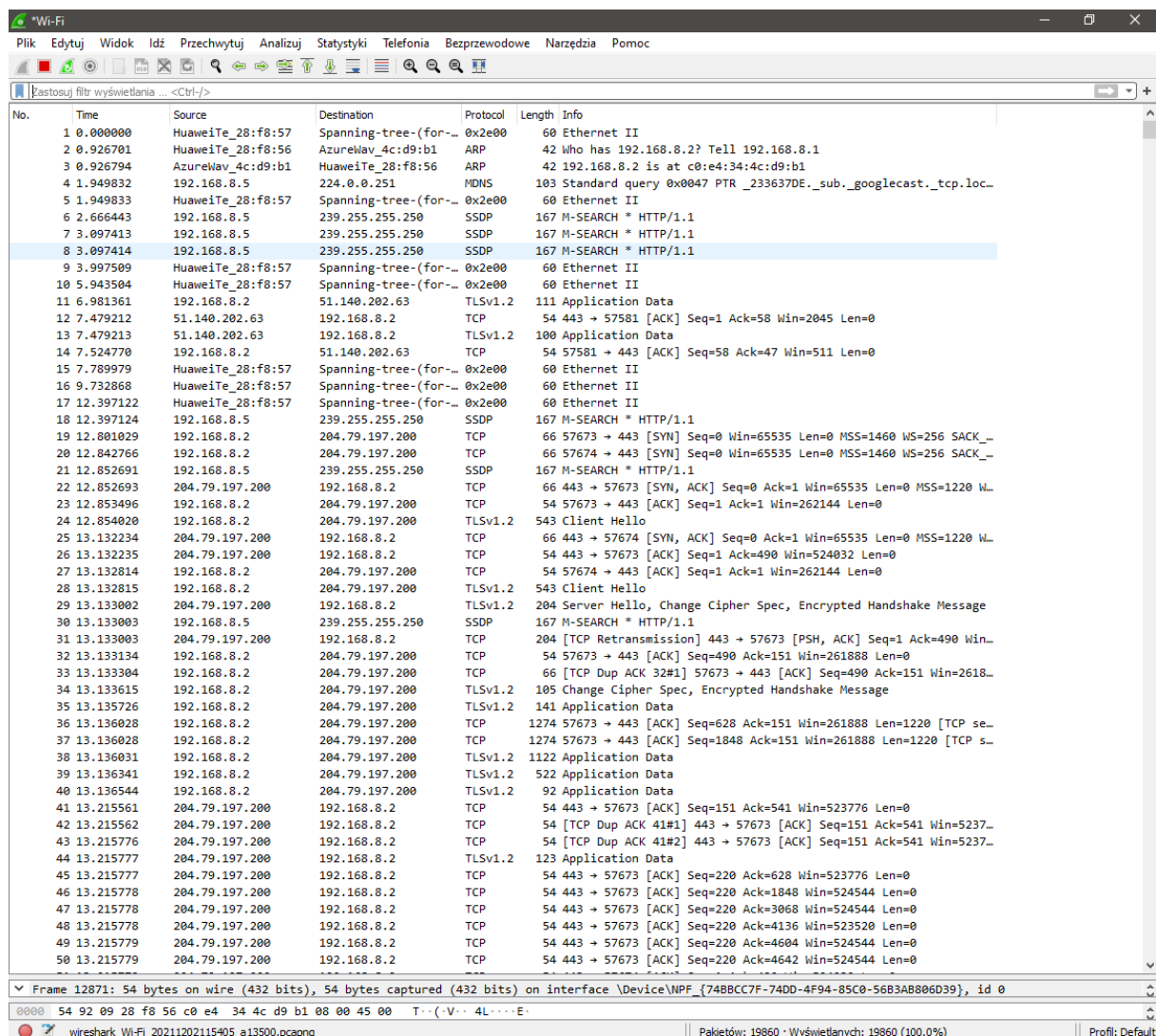
Przechwytywanie ruchu sieciowego można zdefiniować jako zbieranie pakietów na określonej trasie. Bierne przechwytywanie polega na nieinwazyjnym zbieraniu i czytaniu wszystkiego co wychodzi z naszej karty sieciowej. Przedstawię dwa główne programy wraz z przydatnymi funkcjami i komendami, które posłużą nam do analizy ruchu sieciowego z naszej karty sieciowej, a są to Wireshark oraz TCPDump.



Rysunek 1.11. Bierne przechwytywanie ruchu sieciowego.

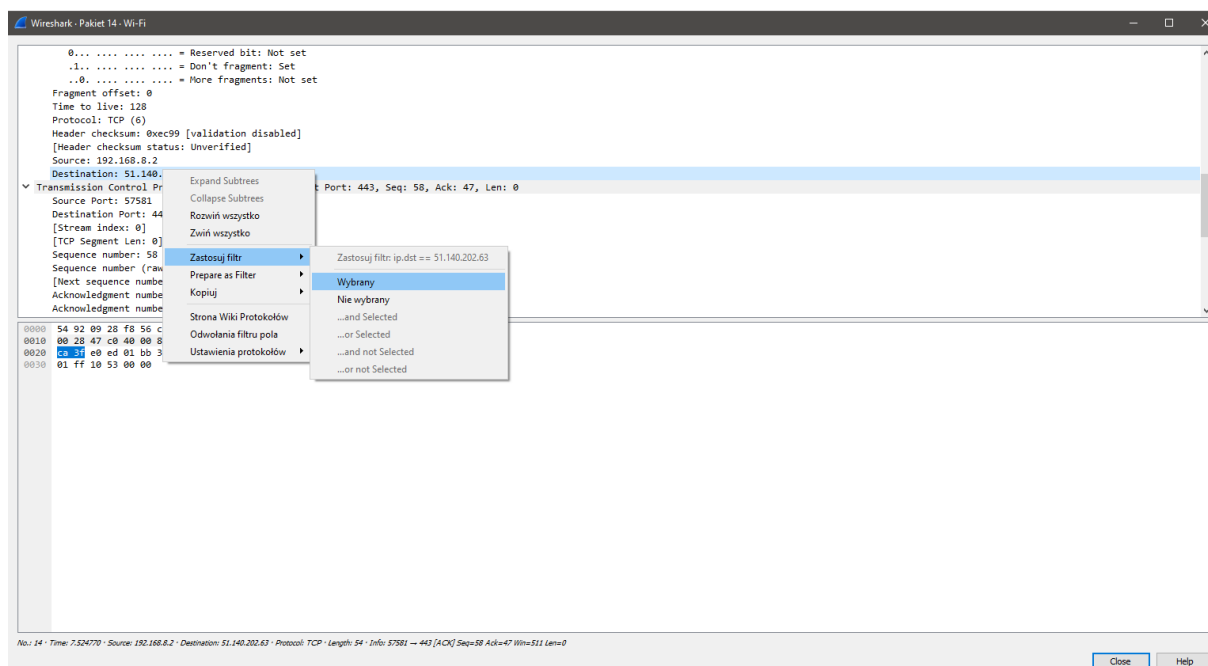
Tak jak pokazuje rysunek 1.11 urządzenie do biernego przechwytywania ruchu sieciowego zbiera informacje pozyskane z karty sieciowej. W jej sieć wpadają wszystkie pakiety nawet te które nie są adresowane do nas, a w trybie rozgłoszeniowym wysyłane do innego hosta w sieci.

Wireshark- to jedno z najbardziej popularnych urządzeń do biernego przechwytywania ruchu sieciowego. Do prawidłowej pracy wymagane jest uruchomienie tego narzędzia jako administrator. Następnie wystarczy wybrać interesujący nas interfejs sieciowy czyli naszą kartę i urządzenie rozpocznie swoją pracę przechwytyjąc podsłuchane pakiety.



Rysunek 1.12. Okno Wireshark.

W oknie Wireshark możemy zobaczyć wszystkie podsłuchane pakiety. Aby wyodrębnić interesujące nas pakiety należy zastosować filtr. Filtr możemy założyć na dwa sposoby albo wpisać go ręcznie, jeśli znamy dokładną konstrukcję np. filtr „ARP || ICMP” który pokaże nam wyłącznie te pakiety lub zaznaczyć interesującą nas część co pokazuje rysunek 1.13.



Rysunek 1.13. Filtr Wireshark.

Najbardziej przydatne filtry wykorzystywane w narzędziu Wireshark przedstawia rysunek poniżej:

Filtr	Opis
ARP	prosty filtr wykorzystujący nazwę protokołu
ip.dst == 51.140.202.63	filtr do wyszukania pakietów o docelowym adresie IP
ip.src == 192.168.8.2	filtr do wyszukania pakietów o źródłowym adresie IP
tcp.dstport == 443	filtr do wyszukania docelowego portu dla protokołu TCP.
tcp.srcport == 443	filtr do wyszukania źródłowego portu dla protokołu TCP.
dns.qry.name == "self.events.data.microsoft.com"	filtr, który pokazuje zapytania DNS o nazwę podaną w cudzysłowie.
dns.resp.name == "self.events.data.microsoft.com"	filtr, który pokazuje odpowiedzi DNS o nazwę podaną w cudzysłowie.
eth.type == 0x0800	pokazuje ramki Ethernet których typ to 0x0800, czyli wiemy, że znajduje się tam IPv4.

Rysunek 1.14. Przydatne filtry w użytkowaniu narzędzia Wireshark.

Oczywiście filtrów jest dużo więcej, lecz znajomość podstawowych ułatwia pracę z programem. Możemy zakładać również filtry złożone wykorzystując podstawowe operatory logiczne takie jak: OR, AND i NOT np. not TCP co spowoduje niewyświetlanie tego protokołu. **TCPDump**- jest to proste narzędzie do pasywnego przechwytywania ruchu sieciowego które otrzymamy na systemach typu UNIX, czyli Linux, Solaris, FreeBSD itp. Jest to konsolowe, czyli bez wyświetlacza graficznego które umożliwia również zapis do pliku zebranych danych. Do obsługi programu warto więc znać podstawowe komendy oraz zaznajomić się z terminalem chociażby w systemie Linux. W celu zainstalowania programu TCPDump w systemach typu UNIX należy zapoznać się z podstawowymi komendami trzeba pamiętać, że do wykonania instalacji i do używania potrzebujemy uprawnień administratora. W celu pokazania zasady działania tego narzędzia skorzystam z popularnego systemu Linux w wersji Ubuntu. Jeśli nie posiadamy TCPDump należy je zainstalować dopiero po instalacji możemy używać tego narzędzia, ponieważ nie jest one domyślnie instalowane. Najważniejsze polecenia w pracy z programem TCPDump przedstawia poniższy rysunek:

Polecenie	Opis
<code>sudo apt-get install tcpdump</code>	służy do instalacji TCPDump
<code>sudo man tcpdump</code>	otwiera podręcznik systemu linux dla polecenia tcpdump
<code>tcpdump -h</code>	wyświetla tzw. helpa, czyli pomoc
<code>sudo tcpdump -D</code>	pokazuje dostępne interfejsy sieciowe
<code>sudo tcpdump -c 5</code>	przechwytuje tylko do 5 pakietów
<code>sudo tcpdump arp</code>	przechwytuje tylko pakiety arp
<code>sudo tcpdump ''dst port 80''</code>	przechwytuje ruch na porcie 80
<code>ping -c4 192.168.4.1 & sudo tcpdump icmp</code>	przechwytuje pakiety ICMP wygenerowane poleceniem ping
<code>sudo tcpdump -w pakiety.dmp</code>	zapisze przechwycone pakiety w formacie dmp
<code>sudo tcpdump -r pakiety.dmp</code>	odczyta zapisane pakiety

Rysunek 1.15. Najważniejsze komendy programu TCPDump.

12:23:55.684760	IP	xx-fbcdn-shv-01-frt3.fbcdn.net.https	>	192.168.8.2.55362:	UDP,	length 1232
zaczek czas		adres źródłowy		adres docelowy		długość
protokół sieciowy				protokół transportowy		

Rysunek 1.14 Pakiet w programie TCPDump wraz z opisem kolumn.

Rysunek 1.16. Analiza budowy pakietu TCPDump.

Do śledzenia ruchu sieciowego możemy również wykorzystać programy które pokażą nam wszystkie procesy odpowiadające za połączenia sieciowe. Otóż, gdy aplikacja chce połączyć się z serwerem za pomocą protokołu TCP, musi nastąpić wymiana trójetapowa tak zwany uścisk dłoni. System do obsługi każdej aplikacji na przykład podczas której wykorzystujemy trójetapową wymianę wykorzystuje procesy. Śledzenie procesów sieciowych może dostarczyć cennych informacji dotyczących ruchu sieciowego w sieci. Podczas obserwacji za pomocą nawet darmowego programu możemy zauważyć, że nawet programy które z pozoru nie korzystają z połączenia sieciowego mają otwarte porty i nasłuchują co stwarza dodatkowe zagrożenie dla naszego komputera. Patrząc na rysunek 1.17. możemy dostrzec, że nawet programy z pozoru przyjazne takie jak antywirus nasłuchują na lokalnych portach. Nawet program Skype który jest nieużywany, a jest zainstalowany na komputerze może się wydawać, że nie powinien korzystać z zasobów sieciowych. Jak widać program ten ma ustanowione połączenie sieciowe bez wiedzy użytkownika co stwarza podejrzenie, że może wysyłać dane bez naszej wiedzy i może być idealnym przykładem bardzo trudnego do wykrycia wirusa typu Koń Trojański. Sytuacja ta zmusza do refleksji, ponieważ świetnie pokazuje, że nie tylko same pakiety należy analizować, a także co najważniejsze ich źródło w naszym komputerze.

to numer, który jest przyznawany każdemu procesowi. PID jest numerem unikalnym co oznacza, że żadne dwa procesy nie będą miały przypisanego takiego samego numeru PID. Wykorzystując polecenie netstat oraz polecenie tasklist możemy otrzymać wynik podobny do wcześniej użytego programu, który pokazywał to w sposób zwięzły i przystępny dla użytkownika.

WybierzAdministrator: Wiersz polecenia
C:\WINDOWS\system32>tasklist

Image Name	PID	Session Name	Session#	Mem Usage
System Idle Process	0	Services	0	8 K
System	4	Services	0	9 496 K
Registry	124	Services	0	65 412 K
smss.exe	576	Services	0	516 K
csrss.exe	852	Services	0	2 592 K
wininit.exe	940	Services	0	2 768 K
services.exe	652	Services	0	8 248 K
lsass.exe	880	Services	0	20 616 K
svchost.exe	1128	Services	0	48 592 K
fontdrvhost.exe	1172	Services	0	528 K
WUDFHost.exe	1232	Services	0	8 776 K
svchost.exe	1296	Services	0	23 272 K
svchost.exe	1344	Services	0	5 164 K
svchost.exe	1600	Services	0	5 836 K
svchost.exe	1608	Services	0	5 564 K
svchost.exe	1644	Services	0	12 240 K
svchost.exe	1708	Services	0	7 332 K
svchost.exe	1728	Services	0	4 032 K
svchost.exe	1740	Services	0	2 336 K
svchost.exe	1816	Services	0	4 520 K
svchost.exe	1892	Services	0	13 380 K
svchost.exe	1984	Services	0	4 460 K
svchost.exe	2008	Services	0	9 004 K
IntelCpHeciSvc.exe	2036	Services	0	1 560 K
svchost.exe	1404	Services	0	4 680 K
svchost.exe	2192	Services	0	8 028 K
svchost.exe	2324	Services	0	4 032 K
svchost.exe	2432	Services	0	2 868 K
svchost.exe	2464	Services	0	7 008 K
svchost.exe	2472	Services	0	1 452 K
wsc_proxy.exe	2500	Services	0	6 756 K
svchost.exe	2592	Services	0	8 636 K
svchost.exe	2600	Services	0	6 840 K
Memory Compression	2628	Services	0	147 220 K
svchost.exe	2684	Services	0	19 828 K
svchost.exe	2780	Services	0	6 536 K
svchost.exe	2948	Services	0	2 908 K
svchost.exe	2956	Services	0	5 908 K
svchost.exe	3064	Services	0	10 944 K
svchost.exe	2844	Services	0	6 676 K
svchost.exe	2888	Services	0	2 680 K
svchost.exe	3040	Services	0	4 792 K
svchost.exe	3204	Services	0	4 032 K
svchost.exe	3280	Services	0	4 112 K
svchost.exe	3312	Services	0	10 400 K
svchost.exe	3420	Services	0	6 728 K
AVGSvc.exe	3484	Services	0	175 328 K
avgToolsSvc.exe	3868	Services	0	52 520 K
spoolsv.exe	4048	Services	0	9 684 K
svchost.exe	4072	Services	0	9 676 K
svchost.exe	3660	Services	0	2 680 K
svchost.exe	4172	Services	0	13 092 K
armsvc.exe	4200	Services	0	864 K
svchost.exe	4208	Services	0	12 712 K
svchost.exe	4236	Services	0	45 124 K
OneApp.IGCC.WinService.ex	4252	Services	0	7 108 K
ICESoundService64.exe	4264	Services	0	2 684 K
svchost.exe	4304	Services	0	29 280 K
svchost.exe	4344	Services	0	1 240 K

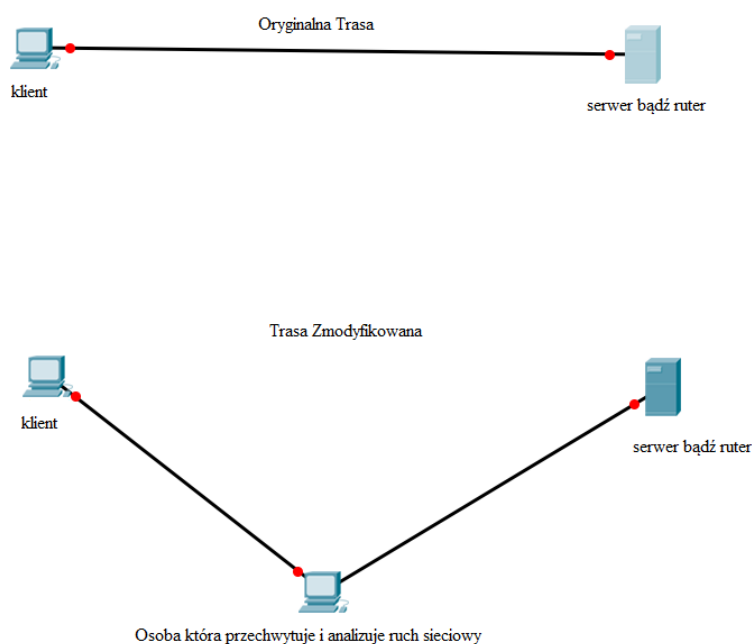
Rysunek 1.19. Polecenie Tasklist.

Przechwytywanie bierne ruchu sieciowego jest ważnym aspektem bezpieczeństwa, ponieważ w 99% przypadkach pomiędzy hakerem, a ofiarą jest sieć. Dlaczego 99% ponieważ jeszcze jest możliwość, że ktoś w niepowołany nie przez sieć sposób uzyskał dostęp do naszego komputera np. ukradł albo włamał się do naszego pomieszczenia. Wyłączając te przypadki jedynym możliwym dostępem oprócz fizycznego dostępu jest dostęp zdalny, a więc analiza ruchu sieciowego dostarcza nam informacji o działaniu aplikacji i pozwala wykryć wszystko co może być niebezpieczne. Dzięki tym działaniom możemy zapewnić na najwyższym poziomie ochronę przed niebezpieczeństwem czyhającym na nas w sieci. Ważnym poleceniem, któremu warto się przyjrzeć jest polecenie Tracert które pokazuje nam trasę pakietu co jest ważnym

elementem przy wykrywaniu zmian na trasie pakietu, siłę i znaczenie dla wiarygodności transmisji bitowej jest szczególne co uwidocznię w następnym podrozdziale.

1.4 Aktywne przechwytywanie ruchu sieciowego.

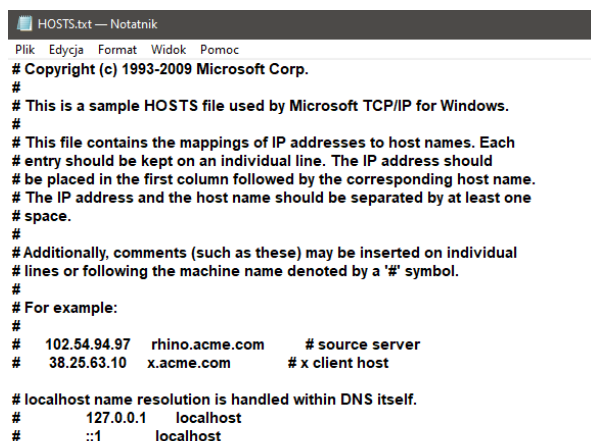
Odmiernym sposobem na przechwytywanie ruchu sieciowego które już w sposób zaawansowany pozwala na jego analizę oraz jest formą ingerencji w ruch pakietów w sieci to aktywne przechwytywanie, czyli po prostu atak typu MiTM atak z człowiekiem pośrodku. Trasa naszego pakietu ulega zmianie, ponieważ musimy umieścić nasze urządzenie przechwytyjące pomiędzy nadawcą, a miejscem docelowym. Dobrze tą sytuację ilustruje to rysunek 1.20.



Rysunek 1.20 Aktywne przechwytywanie ruchu sieciowego.

Każdy pakiet w dużym uproszczeniu, jeśli mamy na myśli transmisję w sieci WAN wędruje od nadawcy do odbiorcy. Gdy znajdzie się ktoś kto zaburzy tą transmisję czasami w sposób oczekiwany dla wszystkich np. w celu ustalenia co zamierza niebezpieczny przestępca albo podczas symulacji ataku w celu sprawdzenia mechanizmów obronnych przed tego typami ataku. To mamy do czynienia z wcześniej wspomnianym atakiem MiTM. Atak z urządzeniem, które występuje pomiędzy ustalonymi hostami w sieci jest trudny do wychwycenia dla zwykłego użytkownika. Lecz znajomość prostych komend i częste śledzenie trasy oraz zbieranie informacji o niej daje możliwość wychwycenia nieprawidłowości na docelowej trasie

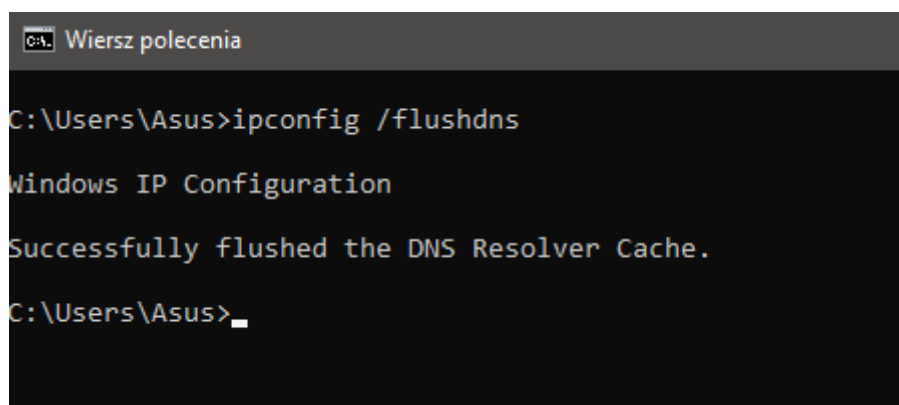
pakietów. Proste polecenie Tracert występujące na systemach z rodziny Windows oraz polecenie Traceroute w rodzinie Linux daje możliwość do monitorowania trasy naszych pakietów co może dać nawet zwykłemu użytkownikowi mechanizm może nie obrony, ale odpowiedniego reagowania i działania, że coś jest nie tak z naszą trasą pakietów w sieci. Atak ten można rozwijać na wiele sposobów, lecz słowo kluczowe to przekierowywanie, które będzie wykorzystywane w każdym sposobie. Pierwszy sposób zrealizowania to przekierowywanie pakietów do serwera proxy który będzie naszym pośrednikiem pomiędzy klientem, a serwerem bądź ruterem. Pierwszy serwer proxy będzie wykorzystywał element przekierowywania portów. Nasze proxy będzie nasłuchiwało na lokalnym adresie 127.0.0.1 do niego przekieruje ruch z wybranym serwerem za pomocą pliku host. Gdy dany ruch zostanie przekierowany to na porcie, który zostanie określony przed włączeniem serwera np. 80 nasze proxy przechwyci i połączy się z adresem docelowym na porcie np. 8080. Ważnym chodź zapomnianym elementem naszego komputera jest plik hosts który wykorzystam właśnie do tego celu. Plik hosts można powiedzieć to dziadek naszego DNS-u. Ponieważ przed uruchomieniem dynamicznego odwzorowywania nazw to plik hosts pełnił rolę zamiany. Mechanizm działania był prosty wpisujemy nazwę komputer jej nie zna idzie do pliku hosts i podmienia wskazaną nazwę na adres IP. Jest to prosta metoda, lecz posiada graniczenia. Otóż sieć rozbudowywała się o kolejne serwery które musiały posiadać własną nazwę. Pliki hosts musiały być aktualizowane cyklicznie więc co noc nazwy te były czerpane z centralnego serwera, a rozmiary ich rosły więc przeszukiwanie stawało się coraz wolniejsze efektywność tego rozwiązania zaczęła spadać. Wtedy to narodził się pomysł, aby zastosować dynamiczne rozwiązywanie nazw tzw. DNS. Lecz plik hosts nadal istnieje w naszych komputerach w katalogu etc nie tylko z systemem Windows, a także w systemach Mac, Linux. Podczas wyszukiwania nasz system najpierw przeszukuje pamięć podręczną, następnie plik hosts, a dopiero na końcu wykonuje zapytanie. Plik hosts ma ogromne możliwości nie tylko przyspieszenie działania niektórych aplikacji, ale także nie jest możliwy atak podczas zapytania DNS w celu przekierowania do innego adresu. Dzięki temu możemy mieć pewność, że cały czas mamy dostęp do właściwego adresu IP i dane podajemy właściwej stronie. Jest też możliwy inny sposób na wykorzystanie tego pliku, a mianowicie działanie, które wykorzystuje zaletę jak i słabość. Za pomocą tego pliku możemy bez problemu dodać własne odwzorowanie i sprawić, aby poprzez zatrucie umożliwić łączenie z naszą fałszywą stroną www. Dlatego należy monitorować jego zawartość, ponieważ każda aplikacja instalowana której dajemy uprawnienia administratora naszego systemu może spowodować niechcianą zmianę. Dlatego należy uważnie dawać uprawnienia naszym aplikacjom. Wygląd w pliku hosts wygląda tak:



```
HOSTS.txt - Notatnik
Plik  Edycja  Format  Widok  Pomoc
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
# 102.54.94.97  rhino.acme.com    # source server
# 38.25.63.10   x.acme.com       # x client host
#
# localhost name resolution is handled within DNS itself.
#          127.0.0.1    localhost
#           ::1        localhost
```

Rysunek 1.21 Plik hosts.

Podczas realizacji tego zadania wykorzystam fakt, że mimo plik hosts jest nie używany to nadal jest przeglądany podczas dynamicznego ustalania nazw. Gdy wpisujemy nazwę domenową np. facebook lub inną to komputer po sprawdzeniu pamięci przeszukuje plik hosts tam umieszczamy wpis tak żeby interesująca nas strona była kojarzona z adresem pętli zwrotnej 127.0.0.1. Teraz należy ustawić serwer proxy który na określonym porcie oraz adresie pętli zwrotnej będzie nasłuchiwał dodatkowo do serwera proxy podajemy adres oryginalnej strony tak żeby komunikacja z tą stroną, mimo że przez serwer pośredniczący przebiegała bez zakłóceń. Serwer będzie nasłuchiwał treść wszystkich pakietów i zapisywał w pamięci co później zostanie wyświetlone na ekranie aplikacji serwera. W pierwszym kroku należy opróżnić pamięć podręczną nazw domenowych w naszym komputerze tak aby dana nazwa nie znajdowała się w pamięci, ponieważ cała operacja nie powiedzie się, jeśli komputer odszuka daną nazwę. Jeśli nazwa nie znajduje się w pamięci wtedy jest przeszukiwany plik hosts. W celu opróżnienia pamięci należy zastosować polecenie `ipconfig /flushdns` w wierszu polecenia co zostało przedstawione na rysunku 1.22.



```
C:\Users\Asus>ipconfig /flushdns

Windows IP Configuration

Successfully flushed the DNS Resolver Cache.

C:\Users\Asus>
```

Rysunek 1.22. Polecenie do czyszczenia pamięci DNS.

[illegible]

Następnie należy skonfigurować proxy to znaczy podać port do nasłuchu oraz adres IP zewnętrznego serwera. Po wykonaniu konfiguracji serwera proxy zostanie utworzony nasłuch na adresie 127.0.0.1 oraz porcie wskazanym na początku np. 80. Cała konfiguracja została pokazana na rysunku 1.21 przy pomocy własnego utworzonego serwera proxy które napisałem w języku C#. Gdy konfiguracja przebiegnie pomyślnie po wpisaniu w okno wyszukiwarki naszej nazwy i odwiedzeniu strony internetowej możemy zatrzymać serwer, który wypisze wszystkie zebrane pakiety, wynik możemy zobaczyć na rysunku powyżej. Jak to działa otóż po wyczyszczeniu nazw domenowych pamięć DNS staje się pusta co wymusza wyszukiwanie nazw domenowych najpierw w pliku hosts następnie za pomocą protokołu DNS. Gdy system otrzyma wynik z pliku hosts uważa to za rzetelną informację i na tej podstawie buduje informację którą wyśle poprzez sieć. Następnie, gdy zaczniemy wyszukiwać daną nazwę na adresie pętli zwrotnej na określonym porcie nasz serwer proxy odezwie się i dalej prześle pakiet do właściwego adresu. Po otrzymaniu odpowiedzi serwer zwraca ją, a w oknie przeglądarki pojawia się wynik. Całość trwa bardzo szybko dzięki czemu nie można zauważyć żadnego opóźnienia co daje złudzenie, że łączymy się z serwerem właściwym i wszystko jest tak jak wcześniej. Jak każdy sposób ma swoje wady i zalety na pewno ten sposób jest dość prosty w zrealizowaniu jak i mechanizm jego działania nie jest skomplikowany. Podczas używania tego typu serwera pośredniczącego nie są wymagane żadne dodatkowe rzeczy np. odpowiednia aplikacja obsługująca, dodatkowy protokół jedyne co potrzebujemy to umiejętność

dokonywania zmian w pliku hosts. Jedną z największych zalet jest prostota, lecz jak wszystko ma swoje wady. Jeśli aplikacja wymaga do poprawnego działania użycia dwóch portów to do poprawnego działania tego należy użyć dwóch proxy jedno dla portu X drugie dla portu X1. Drugim problemem jest to, gdy zajdzie potrzeba połączenia się z drugim serwerem tu też napotkamy problem, ponieważ zaistnieje potrzeba zmiany portu na drugim bądź pierwszym połączeniu. Proxy z przekierowywaniem portu to nie jedyna odmiana serwera proxy do wykorzystania mamy proxy także oparte na protokołach SOCKS dostępne w wersjach 4, 4a i 5, HTTP i HTTPS. Następnym sposobem zrealizowania aktywnego przechwytywania ruchu sieciowego jest wykorzystanie protokołu ARP. Protokół ARP służy do ustalania adresu sprzętowego na podstawie adresu IP. Mechanizm działania protokołu jest prosty, gdy host chce wysłać pakiet do drugiego hosta to potrzebuje kilku bardzo ważnych informacji: znać docelowy adres IP na który zamierza wysłać dane, wiedzieć czy docelowe urządzenie sieciowe jest w tym samym segmencie sieci co host, do którego zamierzamy wysłać dane, znać jaki jest adres sprzętowy urządzenia, do którego zamierzamy wysłać dane. Gdy urządzenie źródłowe zna adres IP drugiego urządzenia to potrafimy odpowiedzieć na dwa powyższe pytania, lecz sam adres IP to nie wszystko potrzebujemy adresu MAC. Uzyskiwanie adresu MAC za pomocą protokołu ARP wygląda następująco, komputer wysyła pakiet żądanie, gdzie adres MAC odbiorcy jest w formie rozgłoszeniowej ff:ff:ff:ff:ff:ff co sprawia, że pakiet jest odbierany przez wszystkie urządzenia, które go usłyszą, lecz tylko urządzenie z określonym adresem IP na nie odpowiada i wysyła swój MAC adres. Ta sytuacja wygląda tak jakbyśmy weszli do domu kolegi wiemy, gdzie on mieszka i chcemy, aby wyszedł tylko on, a nie jego brat lub ktoś inny, krzyczymy stojąc przed domem Franek choć na dwór każdy słyszy nawet rodzice Franka, lecz przybiega tylko Franek i mówi, że jest. Odpowiedź urządzenia nie jest wysyłana w sposób rozgłoszeniowy tylko trafia do jednego urządzenia nadawcy tego żądania dlatego w podanym przykładzie Franek przybiega i mówi, że jest co sprawia, że nikt tego już nie słyszy tylko host, który wysłał żądanie, aby Franek zszedł. Pakiet żądania przedstawia rysunek 1.24, a odpowiedzi 1.25.

arp.pcapng

Plik Edytuj Widok Idź Przechwytyj Analizuj Statystyki Telefonnia Bezprzewodowe Narzędzia Pomoc

arp

No.	Time	Source	Destination	Protocol	Length	Info
62	56.573057	AzureWav_4c:d9:b1	HuaweiTe_28:f8:56	ARP	42	192.168.8.2 is at c0:e4:34:4c:d9:b1
118	75.303985	AzureWav_4c:d9:b1	Broadcast	ARP	42	Who has 192.168.8.1? Tell 192.168.8.2
119	75.311628	HuaweiTe_28:f8:56	AzureWav_4c:d9:b1	ARP	42	192.168.8.1 is at 54:92:09:28:f8:56
125	81.325136	AzureWav_4c:d9:b1	Broadcast	ARP	42	Who has 192.168.8.1? Tell 192.168.8.2
126	81.332182	HuaweiTe_28:f8:56	AzureWav_4c:d9:b1	ARP	42	192.168.8.1 is at 54:92:09:28:f8:56
148	83.336597	AzureWav_4c:d9:b1	Broadcast	ARP	42	Who has 192.168.8.1? Tell 192.168.8.2
149	83.342443	HuaweiTe_28:f8:56	AzureWav_4c:d9:b1	ARP	42	192.168.8.1 is at 54:92:09:28:f8:56
164	96.642202	HuaweiTe_28:f8:56	AzureWav_4c:d9:b1	ARP	42	Who has 192.168.8.2? Tell 192.168.8.1
165	96.642297	AzureWav_4c:d9:b1	HuaweiTe_28:f8:56	ARP	42	192.168.8.2 is at c0:e4:34:4c:d9:b1
172	101.448704	AzureWav_4c:d9:b1	Broadcast	ARP	42	Who has 192.168.8.1? Tell 192.168.8.2
173	101.459242	HuaweiTe_28:f8:56	AzureWav_4c:d9:b1	ARP	42	192.168.8.1 is at 54:92:09:28:f8:56

> Frame 148: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\NPF_{748BCC7F-74DD-4F94-85C0-56B3AB806D39}, id 0

> Ethernet II, Src: AzureWav_4c:d9:b1 (c0:e4:34:4c:d9:b1), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

▼ Address Resolution Protocol (request)

Hardware type: Ethernet (1)

Protocol type: IPv4 (0x0800)

Hardware size: 6

Protocol size: 4

Opcode: request (1)

Sender MAC address: AzureWav_4c:d9:b1 (c0:e4:34:4c:d9:b1)

Sender IP address: 192.168.8.2

Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)

Target IP address: 192.168.8.1

Rysunek 1.24. Żądanie ARP.

arp.pcapng

Plik Edytuj Widok Idź Przechwytyj Analizuj Statystyki Telefonnia Bezprzewodowe Narzędzia Pomoc

arp

No.	Time	Source	Destination	Protocol	Length	Info
62	56.573057	AzureWav_4c:d9:b1	HuaweiTe_28:f8:56	ARP	42	192.168.8.2 is at c0:e4:34:4c:d9:b1
118	75.303985	AzureWav_4c:d9:b1	Broadcast	ARP	42	Who has 192.168.8.1? Tell 192.168.8.2
119	75.311628	HuaweiTe_28:f8:56	AzureWav_4c:d9:b1	ARP	42	192.168.8.1 is at 54:92:09:28:f8:56
125	81.325136	AzureWav_4c:d9:b1	Broadcast	ARP	42	Who has 192.168.8.1? Tell 192.168.8.2
126	81.332182	HuaweiTe_28:f8:56	AzureWav_4c:d9:b1	ARP	42	192.168.8.1 is at 54:92:09:28:f8:56
148	83.336597	AzureWav_4c:d9:b1	Broadcast	ARP	42	Who has 192.168.8.1? Tell 192.168.8.2
149	83.342443	HuaweiTe_28:f8:56	AzureWav_4c:d9:b1	ARP	42	192.168.8.1 is at 54:92:09:28:f8:56
164	96.642202	HuaweiTe_28:f8:56	AzureWav_4c:d9:b1	ARP	42	Who has 192.168.8.2? Tell 192.168.8.1
165	96.642297	AzureWav_4c:d9:b1	HuaweiTe_28:f8:56	ARP	42	192.168.8.2 is at c0:e4:34:4c:d9:b1
172	101.448704	AzureWav_4c:d9:b1	Broadcast	ARP	42	Who has 192.168.8.1? Tell 192.168.8.2
173	101.459242	HuaweiTe_28:f8:56	AzureWav_4c:d9:b1	ARP	42	192.168.8.1 is at 54:92:09:28:f8:56

> Frame 149: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\NPF_{748BCC7F-74DD-4F94-85C0-56B3AB806D39}, id 0

> Ethernet II, Src: HuaweiTe_28:f8:56 (54:92:09:28:f8:56), Dst: AzureWav_4c:d9:b1 (c0:e4:34:4c:d9:b1)

▼ Address Resolution Protocol (reply)

Hardware type: Ethernet (1)

Protocol type: IPv4 (0x0800)

Hardware size: 6

Protocol size: 4

Opcode: reply (2)

Sender MAC address: HuaweiTe_28:f8:56 (54:92:09:28:f8:56)

Sender IP address: 192.168.8.1

Target MAC address: AzureWav_4c:d9:b1 (c0:e4:34:4c:d9:b1)

Target IP address: 192.168.8.2

Rysunek 1.25. Odpowiedź ARP.

Jak system uzyska odpowiedź o docelowym adresie MAC będzie mógł poprawnie skonfigurować pakiet oraz zapisać go w tablicy ARP. Tablica ARP zbudowana jest z 3 kolumn: pierwsza zawiera adres IP, druga adres fizyczny, trzecia typ wpisu. Wpis może być dynamiczny lub statyczny. Styczny jest dodawany ręcznie poprzez wywołanie polecenia `arp -s [adres ip] [adres MAC]` w konsoli, natomiast dynamiczny jest dodawany właśnie poprzez wykorzystanie protokołu ARP. Protokół ARP nie gwarantuje, że dane zapytanie bądź odpowiedź zawiera wiarygodne dane jak i nie sprawdza od kogo pochodzą te dane daje możliwość również odbierania odpowiedzi ARP nawet jeśli żądanie nie zostało nigdy wcześniej wysłane co

pozwała na skuteczne podszywanie się pod określone urządzenie sieciowe. Podczas ataku, atakujący zatruwa tablice ofiary jak i rutera tak żeby pakiety swobodnie były przesyłane przez komputer, który dokonuje ataku. W wyniku ataku w komputerze ofiary tablica zawiera powiązany adres IP rutera z adresem MAC ofiary dzięki czemu komunikacja za pomocą warstwy drugiej jest nieprawidłowa, a pakiety wędrują do niewłaściwej maszyny. Atak ten można przeprowadzić za pomocą programu Ettercap. Do wykonania ataku musimy znać adres ofiary i rutera oraz posiadać program Ettercap. Program Ettercap jest to wielofunkcyjne urządzenie, które umożliwia nie tylko zatruwanie tablicy ARP, ale dostarcza wielu narzędzi potrzebnych do aktywnego przechwytywania ruchu sieciowego. Żeby wykonać atak MiTM należy najpierw uruchomić Ettercap następnie:

- wybrać opcję Sniff/Unified sniffing,
- wybrać odpowiedni interfejs sieciowy na którym chcemy wykonać atak MiTM,
- teraz należy wyszukać urządzenia sieciowe Host/Scan For Host,
- po dokonaniu skanowania wybrać z listy odpowiednie urządzenia sieciowe Host/Host List,
- wybieramy hosty zaznaczamy adres IP ofiary i dodajemy jako Target 1 następnie jako Target 2 wybieramy adres rutera,
- teraz włączamy infekowanie Mitm/Arp poisoning.

Przed wykonaniem ataku tablica ARP po wykonaniu polecenia arp -a wygląda tak jak na poniższym rysunku:

```
C:\> Wiersz polecenia

Microsoft Windows [Version 10.0.19043.1415]
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\Asus>arp -a

Interface: 192.168.8.2 --- 0xb
  Internet Address      Physical Address      Type
  192.168.8.1           54-92-09-28-f8-56     dynamic
  192.168.8.3           08-00-27-d6-dc-32     dynamic
  192.168.8.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251          01-00-5e-00-00-fb     static
  224.0.0.252          01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 192.168.56.1 --- 0xd
  Internet Address      Physical Address      Type
  192.168.56.255        ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251          01-00-5e-00-00-fb     static
  224.0.0.252          01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static

C:\Users\Asus>
```

Rysunek 1.26. Tablica ARP przed infekcją.

Jako urządzenie atakujące wybrałem system Linux Ubuntu z numerem IP 192.168.8.3 które możemy zauważyć na powyższym rysunku. Należy zwrócić uwagę na adres MAC który różni się od adresu MAC rutera, ale po przeprowadzonym ataku tablica będzie wyglądać tak jak na poniższym rysunku:


```
Wiersz polecenia
Microsoft Windows [Version 10.0.19043.1415]
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\Asus>arp -a

Interface: 192.168.8.2 --- 0xb
  Internet Address      Physical Address      Type
  192.168.8.1           08-00-27-d6-dc-32     dynamic
  192.168.8.3           08-00-27-d6-dc-32     dynamic
  192.168.8.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static

Interface: 192.168.56.1 --- 0xd
  Internet Address      Physical Address      Type
  192.168.56.255        ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static

C:\Users\Asus>
```

Rysunek 1.27. Zainfekowana tablica ARP.

Ruter i urządzenie atakujące mają ten sam adres MAC co sprawia, że komunikacja w warstwie drugiej ISO/OSI bądź pierwszej TCP/IP została zakłócona i teraz pakiety będą wędrowały do najpierw do urządzenia oznaczonego adresem 192.168.8.3, a dopiero później do rutera, jeśli urządzenie przekaże pakiety do niego. Atak ten jest trudny do wykrycia, ponieważ oprócz zainfekowanej tablicy nic nie ukazuje nam, że jesteśmy zaatakowani bądź trasa pakietów została zmieniona i na tej drodze istnieje inne urządzenie, które chce odczytywać nasze pakiety. Przy unikaniu zagrożeń należy monitorować wszelkie parametry systemowe dzięki czemu możemy mieć dokładny podgląd na to czy nasz komputer jest w pełni bezpieczny, ponieważ nawet jeśli my zadamy o to, aby nasz komputer nie został zainfekowany to nie możemy bronić się przed tym co jest na zewnątrz. Po wyjściu pakietów z naszego komputera nie możemy być pewni czy nasze pakiety nie zostaną przechwycone i podmienione. Jeśli nie można wejść do garażu by zabrać samochód wystarczy, że ktoś wyjedzie nim na drogę i zostawi na parkingu.

Tak samo jest z siecią nawet jeśli nasza będzie zabezpieczona na wszelki możliwy sposób to wyjście pakietu poza sieć sprawia, że nic nie możemy zrobić, dlatego oprócz zabezpieczenia naszej sieci musimy w rozważny sposób korzystać z sieci internetowej zwykle polecenie tracert informujące o trasie pakietów może być poleceniem, które ustrzeże nas przed nie właściwym dostępem do naszych danych. Regularna analiza trasy pakietów i wszelkie zmiany w niej powinny być dla nas sygnałem, że może dziać się coś niedobrego oraz zalecane jest zachowanie szczególnej ostrożności.

1.5 Podsumowanie.

Analiza ruchu sieciowego wymaga odpowiedniego zapoznania się z budową i zasadą działania najważniejszych protokołów sieciowych. W tym rozdziale poddałem analizie budowę zasadę działania protokołu IPv4 i IPv6 które są jednymi z najważniejszych podczas pracy z sieciami komputerowymi. Zapewniają optymalną kontrolę pomiędzy warstwami niskiego i wysokiego poziomu, gdzie istnieje najwięcej rozwiązań sieciowych. Nie można zapominać także o innych protokołach które budują sieć takimi jak np. ARP które wykorzystałem do symulacji ataku z człowiekiem pomiędzy nadawcą a odbiorcą tzw. MiTM. Protokół ARP, mimo że może posłużyć do ataku pełni ważną funkcję, ponieważ znajduje odpowiadające adresy fizyczne przy wykorzystaniu adresów logicznych co jest ważne w identyfikacji pojedynczego urządzenia. Następnym protokołem, który uważam za jeden z najważniejszych to protokół TCP i UDP które działają w warstwie transportowej. TCP to protokół połączeniowy gwarantuje, że wszystkie wysłane pakiety dotrą do miejsca docelowego zaś UDP to protokół bezpołączeniowy więc nie może nam zagwarantować, że nasze pakiety trafią do miejsca docelowego. Jednak oba są ważne dla działania sieci. Podczas opisywania zasady działania wielu protokołów postanowiłem poszukać informacji oraz samemu poddać je szczegółowej analizie pod względem bezpieczeństwa. Okazało się, że stara zasada, jeśli coś jest zrobione przez człowieka, który nie jest idealny tak i jego rzeczy też nie są psują się zawierają luki. DNS protokół warstwy aplikacji który służy do tłumaczenia adresu IP na nazwy zrozumiałe dla człowieka wydaje się, że nie ma słabości, ponieważ tylko tłumaczy nazwę na adres IP, ale jeśli teraz wpiszę nazwę i ktoś podmieni adres to mamy olbrzymi problem, ponieważ połączymy się z fałszywą stroną, która może wyłudzić od nas cenne informacje na temat naszego konta chociażby na Facebooku lub co gorsza konta bankowego i naszych wrażliwych danych. Czy można się przed tym ustrzec przed atakiem na naszą sieć optymista powie, że można pesymista, że zawsze coś można przeoczyć, a realista powie, że można, jeśli zabezpieczymy wszystkie luki wyłączymy wszelką automatyzację i ograniczymy ruch do minimum wtedy łatwiej jest zabezpieczyć nasze dane. Prostim, a skutecznym sposobem na zabezpieczenie danych jest nie trzymanie ich na

komputerze, który ma dostęp do sieci. Wtedy możemy być pewni, że nasze zdjęcia filmy nie przenikną do Internetu i nikt nie dokona szantażu na nas na podstawie zebranych danych. Dodatkowo warto pamiętać o innym nośniku danych który nigdy nie ma styczności z innymi komputerami które mają dostęp do sieci wyłącznie z kamerą bądź innymi urządzeniami, które są nam niezbędne do zbierania osobistych danych. Ten sposób jest najbardziej skutecznym rozwiązaniem, które możemy używać. Lecz wiele osób nie może pozwolić sobie na taki wydatek warto wtedy po prostu zaopatrzyć się w drugi system na tym samym komputerze, który nie będzie miał dostępu do sieci, a dane na nim zostaną zaszyfrowane co pozwoli w pełni zabezpieczyć nasze wrażliwe informacje. W dalszej części pracy skupiłem się na analizie ruchu sieciowego poprzez wykorzystanie dwóch narzędzi Wireshark oraz TCPDump które są jednymi z najpopularniejszymi aplikacji do analizy ruchu sieciowego w sposób bierny. Możemy bez problemów podsłuchiwać pakiety które trafiają do naszej karty sieciowej dzięki czemu mamy obraz co opuszcza nasz komputer jak i co do niego trafia. Pozwala nam to na sprawdzenie czy aplikacje zainstalowane na naszym komputerze mają ukryte zamiary i czy działają w sposób zamierzony do tego jak deklarował nam producent danej aplikacji bądź recenzent na stronie internetowej z której pobieraliśmy naszą aplikację. Zwróciłem uwagę również na ważny aspekt, a mianowicie na śledzenie procesów sieciowych za pomocą prostego programu jak również pokazałem, jak uzyskać identyczny wynik bez pobierania oprogramowania do śledzenia procesów sieciowych za pomocą polecenia: netstat i tasklist. Procesy sieciowe mogą pokazać to co nie pokażą nam w taki sposób przechwycone pakiety dzięki nim mamy wyobrażenie, które aplikacje mogą zagrażać naszemu systemowi, naszym danym. Jeśli zauważymy, że nie używamy aplikacji, a oczekuje na połączenie bądź nawiązuje połączenie znaczy, że może być potencjalnym trudnym do wykrycia koniem trojańskim. Uważam, że, znajomość nie tylko jak działa sieć, a również nasza jednostka może dać nam szczegółową wiedzę jak zabezpieczyć nasz komputer. Na końcu rozważań pierwszego rozdziału skupiłem się na pokazaniu drugiego bardziej zaawansowanego sposobu przechwytywania ruchu sieciowego jest to aktywne przechwytywanie ruchu sieciowego, czyli tak zwany atak z człowiekiem pośrodku skrót MiTM. Wykorzystałem do tego proxy sieciowe z przekierowywaniem portów oraz program ettercap który umożliwia zatrzymanie tablicy ARP dzięki czemu pakiety zmieniają trasę i poruszają się przez inne urządzenie, a dopiero później są przekazywane do urządzenia sieciowego które wysyła je w dalszą trasę. Warto też w tym miejscu wspomnieć, że również dobrym sposobem na realizację w sieciach w których występuje okablowanie strukturalne na zwykłe fizyczne podłączenie się do przewodu co również może być trudne do wykrycia, jeśli zrobimy to w porze, której nie są użytkowane

systemy bądź w przerwie dzięki czemu nikt nie zauważy, że coś jest nie tak. Każdy skuteczny sposób jest dobry każdy ma wady i zalety, lecz najważniejsze to nie wykorzystywać tego w złej mierze. Trzeba pamiętać, że każdy atak niesie za sobą skutki które mogą być różne. Kiedyś czytałem, że po dokonaniu skanowania sieci drukarki oszalały i zaczęły drukować czarne strony. Dlatego warto uważać, ponieważ nie wiemy, jak zareagują urządzenia sieciowe na atak, ponieważ wiele z nich może mieć zaimplementowaną formę obrony co sprawi, że atak będzie nieskuteczny. Lecz najważniejsze to nie dokonywać ataków w złej mierze tylko po to, aby móc pomagać i chronić, ponieważ jak wiemy najlepszą formą obrony przed zagrożeniem jest atak. W następnym rozdziale skupię się jak na ważnym aspekcie, który dotyczy bezpiecznej i wiarygodnej transmisji danych jakim jest kryptografia. Postaram się w podstawowy sposób opisać najważniejsze jej elementy oraz nawiązać do historii, ponieważ znajomość jej wzbogaca nas o dodatkowe najczęściej proste, lecz skuteczne rozwiązania, które powinniśmy znać.

2 Szyfrowanie i kodowanie danych.

Człowiek, zwierzęta, rośliny chcą czuć się bezpiecznie. Od miliardów lat możemy dostrzec różne metody w jaki sposób tworzyć zasady bezpieczeństwa. Rośliny często stosują różne sposoby takie jak kolce, trucizny albo wybierają miejsca, które są niedostępne dla drapieżników. Zwierzęta często zakopują swoje zdobycze przystosowują się do otoczenia tak samo człowiek stara się zabezpieczyć siebie nie tylko to co posiada, ale również to co myśli pisze czuje itp. Kiedy pojawiła się potrzeba szyfrowania i ukrywania informacji myślę, że wtedy, gdy po raz pierwszy pojawiło się pismo wtedy to człowiek starał się w jakiś sposób ukryć to przed wrogami, ażeby oni nie poznali jego zamiarów. W tym rozdziale moja praca dotyczyła jakże ważnego aspektu wiarygodności przesyłanych danych, a mianowicie pojęcia związane z kryptografią. W głębi tego rozdziału postaram się opisać ważne elementy szyfrowania i kodowania, lecz warto na początku rozróżnić oba terminy więc czym różni się szyfrowanie od kodowania otóż szyfrujemy pojedyncze elementy składowe słowa np. b to =, a kodowanie to jest nie skupianie się na pojedynczych literach, lecz na jednym słowie, które należy w pewny sposób ukryć np. wyraz „Słońce” zamieniamy na „Ala”. Kryptografia była i jest ważnym elementem współczesnego świata. Spotykamy się z nią na każdym możliwym kroku. Gdy wysyłamy wiadomości, korzystamy z banku wysyłamy wiadomości email wszędzie tam jest obecna. Bez kryptografii nie można byłoby korzystać bezpiecznie z tego co znamy dzisiaj szyfrowanie bądź kodowanie jest bardzo ważne, ponieważ uniemożliwia odczytywanie naszych wrażliwych danych. Dlatego podczas korzystania ze stron internetowych ważne jest, aby zwracać uwagę czy połączenie z daną witryną jest szyfrowane. Szyfrowane połączenie zabezpiecza nasze dane przed nieuprawnionym dostępem. Ważne jest, aby podczas transakcji

zakupu na stronach internetowych zwracać uwagę na to czy połączenie z daną witryną było szyfrowane i zabezpieczone tak aby uniemożliwić przeglądanie danych które podajemy podczas zakupów. W tym rozdziale skupię się na przedstawieniu różnych sposobów zabezpieczania informacji od starożytności do teraz. W pierwszym podrozdziale skupię się na aspektach historycznych związanych z kryptografią można powiedzieć, że będzie to wstęp do następnego podrozdziału, gdzie pokaże pierwsze sposoby szyfrowania jakim posługiwali się ludzie. Następnie skupię się na bardzo popularnej maszynie wykorzystywanej podczas II wojny światowej przez Nazistowskie Niemcy do podboju świata, a cały rozdział zakończę na opisanie dzisiejszych sposobów jakie są wykorzystywane przy zabezpieczaniu przesyłanych danych.

2.1 Starożytne początki zabezpieczania informacji.

Pierwsze potwierdzone zapisy dotyczące używania szyfrów pochodzą już z Mezopotamii. Znalezione tam tablice świadczące o używaniu szyfrów pochodzą z około 1500 roku p.n.e. Kilkanaście wieków później II wieku p.n.e. w Grecji został opracowany prosty szyfr oparty o tablicę podstawień, gdzie literę zamieniano na dwucyfrową liczbę autorem tego rozwiązania był historyk Polibiusz. Tablica wyglądała tak jak na poniższym rysunku:

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Rysunek 2.1. Tablica Polibiusza.

Teraz korzystając z takiej tablicy zaszyfrujemy słowo kryptografia w wyniku szyfrowania otrzymamy taki ciąg cyfr 25(k) 42(r) 54(y) 35(p) 44(t) 34(o) 22(g) 42(r) 11(a) 21(f) 24(i) 11(a). Dla utrudnienia warto pisać cyfry obok siebie co utrudnia złamanie informacji, ponieważ nie wiemy, ile cyfr szyfruje pojedynczą liczbę więc osoba, która przechwyci tą informację będzie musiała najpierw pomyśleć, ile cyfr koduje pojedynczą literę alfabetu czy występuje tu jeden wyraz, a może mamy do czynienia z kodowaniem, czyli pojedyncza litera to słowo pewne słowo także nawet prosty szyfr może przysporzyć wiele problemów na które kryptograf musi zwrócić uwagę. Odkodowanie pojedynczej informacji jest zazwyczaj problematyczne, lecz gdy posiadamy więcej danych na temat użytego sposobu to jesteśmy w stanie rozszyfrować co autor próbował przed nami ukryć. Pierwszym znanym urządzeniem wykorzystanym do zaszyfrowania tekstu jest Scytale. Początki istnienia datuje się na V w p.n.e. w Greckiej Sparcie. Jest to kawałek pręta, który mógł mieć różny kształt, a na który nawijano pasek tam pisano

zaszyfrowaną wiadomość po czym odwijano pasek dzięki czemu kurierzy, którzy przesyłali wiadomość mogli wykorzystywać różne dodatkowe techniki ukrycia wiadomości np. owijali się tym paskiem co utrudniało rozpoznanie, że mamy do czynienia z kurierem. Aby odszyfrować należało w taki sam sposób nawinąć na pręt pasek z zaszyfrowaną wiadomością dzięki czemu mogliśmy odczytać wiadomość. Pręt ten oprócz odpowiedniej długości posiadał również odpowiednią grubość dzięki czemu bez niego nie można było odczytać wiadomości, a odszyfrowanie w tamtych czasach było bardzo trudne. Im dłuższa wiadomość tym większa liczba przestawień do otrzymania co dodatkowo utrudniało złamanie szyfru. Jest to dość prosty sposób, lecz łatwy do zastosowania w każdych warunkach mając kawałek tkaniny możemy owinąć ją nawet o własną rękę napisać wiadomość dodać dodatkowe litery na bokach i mamy kawałek szmatki o starym sposobie szyfrowania. Taki sposób może szyfrowania może utrudniać wychwycenie, ponieważ rzadko kto może znać ten sposób jak i nawet domyśleć się, że właśnie w tak banalny sposób otrzymaliśmy szyfr. Dlatego wart znać początkowe rozwiązania, ponieważ zazwyczaj są bardzo proste i niezwykle skuteczne. Następnym ciekawym szyfrem pochodzącym również z basenu Morza Śródziemnego ze starożytnego Rzymu jest szyfr tzw. Cezara od nazwy władcy tamtejszego imperium Gajusza Juliusza Cezara. Szyfr ten datuje się na okres 1 wiek p.n.e. W przeciwieństwie do szyfru przestawieniowego z którym mieliśmy do czynienia przy Scytale szyfr Cezara należy do grupy szyfrów podstawieniowych. Szyfr podstawieniowy jak sama nazwa wskazuje opiera się na podstawieniu jednej litery za drugą lub za kilka liter znaków bądź cyfr np. za A podstawiamy O*. W szyfrze Cezara wykorzystujemy przesunięcie alfabetu o 3 podczas podstawienia co można dostrzec na poniższym rysunku:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	R	S	T	U	V	W	X	Y	Z	alfabet jawny
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	
D	E	F	G	H	I	J	K	L	M	N	O	P	R	S	T	U	V	W	X	Y	Z	A	B	C	alfabet tajny

Rysunek 2.2 Szyfr Cezara.

U góry występuje alfabet jawny który jest powszechnie znany u dołu zaś alfabet tajny który jest znany niewielkiej liczbie osób. Szyfrowanie polega na podstawieniu liter alfabetu jawnego za alfabet tajny co daje taki rezultat: ADAM alfabet jawny = DGDP alfabet tajny. Odkodowanie przebiega w sposób odwrotny do pokazanego to znaczy wiadomość alfabetu tajnego zastępujemy alfabetem jawnym. Ciekawą metodą zabezpieczania informacji jest Steganografia która polega na ukrywaniu poufnych informacji w otwartych wiadomościach, które są powszechnie znane.

Przykładem takiego sposobu jest tzw. niewidzialny atrament, który jest znany od wielu wieków, bo aż od V wieku p.n.e. Podczas wojny Persji ze Spartą narodził się również ciekawy pomysł otóż, gdy dowiedziano się o przygotowywanej ofensywie przeciwko Sparcie jeden z Greków Demaratos ukrył wiadomość w drewnie, a następnie pokrył kawałek drewna woskiem co całkowicie ukryło wiadomość i kurier mógł przewieźć ją bez żadnych problemów. Dzięki czemu Sparta została na czas zawiadomiona o ofensywie wojsk Perskich. Każdy człowiek pragnie czuć się bezpiecznie chce, aby jego dane były niedostępne dla niewłaściwych ludzi stąd narodziła się potrzeba stosowania szyfrowania i kodowania informacji. Mogę zaryzykować twierdzenie, że w każdym domu każdy z nas ma swój mały schowek, w którym przechowuje cenne dla siebie rzeczy. Potrzeba bezpieczeństwa jest naszą cechą naturalną stąd zrodziła się potrzeba ukrywania cennych przedmiotów, myśli, informacji itp. W następnym podrozdziale przejdę do czasów nowożytnych. Opiszę w nim na przykład szyfry które zostały wykorzystane podczas pierwszej wojny światowej oraz podstawowe sposoby łamania zaszyfrowanych informacji. Dzięki czemu będzie można odkryć w jaki sposób zostały one złamane i czytelne dla wroga.

2.2 Podstawowe szyfry.

W powyższym podrozdziale opowiedziałem o początkowych sposobach jakie stosowali ludzie do ukrywania informacji. Opowiedziałem różnice między szyfrowaniem, a kodowaniem. W poprzednim podrozdziale padły dwa hasła szyfry przestawieniowe oraz podstawieniowe. **Szyfry podstawieniowe** opierają się na podstawianiu za oryginalną wiadomość pewnych określonych znaków które kodują daną wiadomość świetnym przykładem był omawiany powyżej szyfr cezara, który jest idealnym rodzajem tego typu szyfru. Ten rodzaj podstawienia, w którym znak alfabetu jawnego posiada określony zamiennik w tekście tajnym nazywamy podstawieniem monoalfabetycznym. W jaki sposób można odszyfrować tę wiadomość? W pierwszym kroku należy na podstawie odpowiednio długiej próbki wywnioskować, że mamy do czynienia z tego typu rodzajem szyfrowania. Następnie należy dokonać analizę częstości języka tego tekstu to znaczy zbadać jakie litery występują najczęściej i na tej podstawie dopasowywać zaszyfrowane elementy. Żeby zbadać które z jaką częstotliwością pojawiają się litery w określonym języku należy wziąć książkę bądź długi tekst i policzyć, ile wystąpień miała każda litera. Po dokonaniu analizy należy posegregować otrzymane wyniki dzięki czemu otrzymamy dokładny pogląd na to jakie litery występują najczęściej w tekście. Drugim sposobem jest przyjęcie, że w danym słowie występuje słowo atak. I teraz szukamy miejsca, w którym pojawia się dwa razy ten sam symbol a na pierwszej i na 3 pozycji oczywiście gdy znajdziemy bardziej skomplikowane słowo, gdzie więcej liter się powtarza jest łatwiej odczytać

wiadomość, lecz oczywiście trudniej je znaleźć. Po znalezieniu takich miejsc należy podstawiać za te symbole i starać się odkrywać kolejne w danym tekście szukając najbardziej prawdopodobnych wyrazów. Najlepszym sposobem jest opierać się na dwóch sposobach jednocześnie dzięki czemu mamy możemy łatwiej odszyfrować wiadomość. Następnym rodzajem szyfrowania, w którym powyższe metody łamania nie działają to **szyfr przestawieniowy**. Szyfr przestawieniowy jak sama nazwa wskazuje przestawia kolejność występowania poszczególnych liter. Oznacza to, że wszystkie litery występują w zaszyfrowanym tekście, lecz w odmiennej kolejności. Jednym ze sposobów na zaszyfrowanie jest ułożenie tekstu w kolumnę o określonej szerokości, a braki w ostatniej kolumnie uzupełniamy losowo wybranymi literami bądź znakami. Szyfrowanie, gdzie przestawiamy kolejność występowania kolumn nazywa się transpozycją kolumnową. Kluczem w tym sposobie szyfrowania może być słowo bądź kawałek tekstu, który nie zawiera powtarzających liter. Przykład tego typu szyfrowania przedstawia rysunek poniżej:

E	D	Y	T	U	J
2	1	6	4	5	3
b	y	ć	a	l	b
o	n	i	e	b	y
ć	f	g	e	d	z

Tekst oryginalny być albo nie być

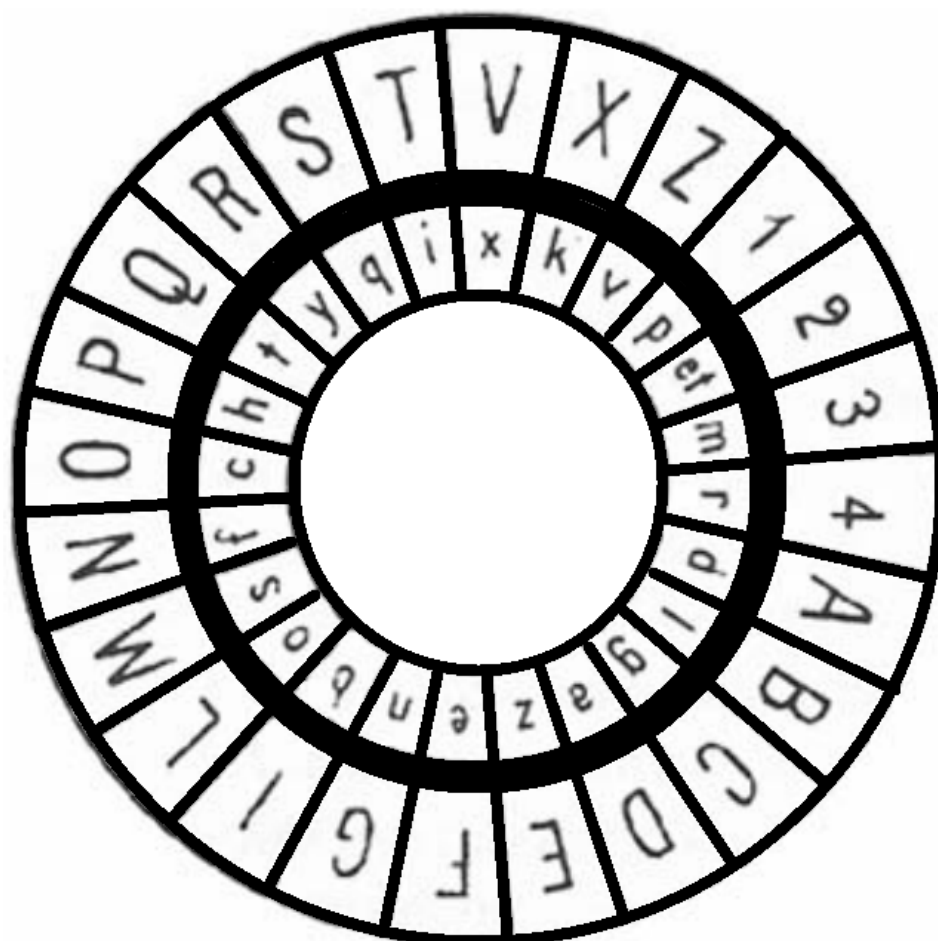
Tekst zaszyfrowany ynfboćbyzaeelbđćig

Rysunek 2.3. Szyfr przestawieniowy.

Aby złamać szyfr przestawieniowy oparty o transpozycję kolumn należy najspierś odnaleźć długość macierzy, gdzie maksymalna długość to liczba liter w tekście. Gdy zostanie to odnalezione należy ustalić prawidłową kolejność np. wzorując się jakie pary liter występują najczęściej w danym języku i tak pogrupować kolumny. W przypadku małej liczby kolumn możemy wypróbować wszystkie kombinacje. Następnie, gdy znajdziemy parę możemy szukać najlepiej pasujące do niej litery tak aby frazy w poszczególnych kolumnach nabierały sensu. Dzięki tym krokom możemy odszyfrować całą wiadomość. Następnym pojęciem będą **szyfry homofoniczne** których początek datuje się na XV wiek. Ten rodzaj szyfrowania powstał w wyniku braku bezpieczeństwa szyfrów podstawieniowych. Do XV wieku szyfry te były głównym sposobem zabezpieczania informacji i nie stosowano innych metod. W związku z rozwojem zainteresowania krypto analizą coraz więcej osób znało sposoby odszyfrowywania

tego typu informacji. Przyczyniło się to, do wyszukania innego sposobu szyfrowania tak aby nie można było odszyfrować tekstu tradycyjnymi metodami opierającymi się o analizę częstości występowania poszczególnych liter w danym języku. Wynikiem tekstu powstał ww. szyfr homofoniczny. Szyfr ten polegał na tym, że dokonywano analizy częstości. Po dokonaniu takiej analizy mieliśmy procentowy obraz z jaką częstością występują litery danego języka. Jeśli po takiej analizie okazało się, że litera „A” występuje z częstością 10% to do jej szyfrowania użyto dziesięć symboli różniących się od siebie dzięki temu zabiegowi każdy znak użyty do szyfrowania występował w takiej samej częstości co pozostałe. Na początku wiele osób myślało, że jest to bariera nie do złamania, lecz to nie prawda nadal można było wykorzystywać cechy danego języka, lecz inne niż dotychczas. Podczas czytania tego tekstu można zauważyć pewne prawidłowości np. że dany wyraz występuje częściej od pozostałych, pewne pary wyrazów również występują częściej od pozostałych to daje możliwość, aby odszyfrować podany tekst. Oczywiście odszyfrowywanie takiego szyfru jest trudniejsze, lecz gdy dysponujemy odpowiednio dużą bazą próbek to możemy odszyfrować taką wiadomość. Podczas odszyfrowywania takiego tekstu należy zwrócić uwagę na występowanie podobnych układów znaków co może sugerować występowanie tych samych wyrazów, lecz w odmiennej konfiguracji dzięki czemu możemy powoli odszyfrowywać jakie pary różnych znaków odpowiadają tej samej literze np. posiadamy zaszyfrowany wyraz adam 11 21 23 25 11 22 23 25 teraz możemy przypuszczać, że para cyfr 22 i 21 odpowiadają tej samej literze. Gdy znajdziemy odpowiedniki danej litery szyfr nadal będzie podatny na złamanie za pomocą tradycyjnej analizy z jaką częstotliwością występuje dana litera w tekście określonego języka. Często podczas szyfrowania wiadomości stosuje się tzw. *puste symbole*. Symbol pusty jest to nieistotny znak, który nie ma żadnego znaczenia przy dekrypcji są one ignorowane. Jednak dla kryptologa są dużym utrudnieniem. Następnym rodzajem z jakim się zapoznam w swojej pracy to szyfry polialfabetyczne. Nazwa ta nie bez przyczyny składa się z dwóch członów, gdzie poli znaczy wiele. Oznacza to, że szyfry polialfabetyczne do swego działania wykorzystują nie jeden alfabet tajny jak to miało miejsce przy monoalfabetycznym, lecz wiele alfabetów tajnych. Każdy skonstruowany alfabet tajny szyfruje inną literę, gdy wykorzystamy wszystkie przygotowane alfabety do szyfrowania wiadomości zaczynamy używać ich znów w tej samej ustalonej kolejności od nowa i tak w kółko aż do zaszyfrowania całej wiadomości. Przykładem takiego szyfru jest XV wieczna Tarcza Albertiego. Zasada działania jest prosta tarcza składa się z dwóch okręgów. Każda z nich zawiera alfabet dodatkowo tarcza zewnętrzna zawiera cztery cyfry. Podczas szyfrowania wiadomości po zaszyfrowaniu jednej litery tarcza wewnętrzna się obracała co w prosty sposób sprawiało, że każda nowa litera była szyfrowana

innym alfabetem. Tarcza zewnętrzna to tzw. alfabet tajny, a tarcza wewnętrzna to alfabet jawny za pomocą tarczy zewnętrznej kodowaliśmy tarczę wewnętrzną. Jedynym utrudnieniem podczas tej metody to, że szyfrant i osoba deszyfrująca musieli posiadać dwa takie same urządzenia oraz znać początkowe ustawienie tarczy. Tarcza została przedstawiona na poniższym rysunku.



Rysunek 2.4. Tarcza Albertiego.

Kolejny sposób szyfrowania który wykorzystywał wiele alfabetów do kodowania pojedynczej wiadomości stworzył Johannes Trithemius. Johannes Trithemius pochodził z Niemiec urodził się 2 lutego 1462 roku. Był niezwykle inteligentnym człowiekiem co pozwoliło mu na udoskonalenie wykorzystania szyfru polialfabetycznego. Stworzył tabele alfabetów służących do kodowania wiadomości. Pierwszy wiersz tabeli zawierał alfabet zaczynający się standardowo od A lecz następny wiersz to przesunięcie alfabetu wyjściowego najpierw o jeden w lewo następnie inkrementowano i przesuwano o dwa itd. Dzięki temu długość i szerokość

tabeli odpowiadała długości alfabetu. Tabele stworzoną przez Johanna Trithemiusa przedstawia poniższy rysunek.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
2	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
3	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
4	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
5	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
6	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
7	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
8	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
9	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
10	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
11	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
12	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
13	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
14	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
15	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
16	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
17	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
18	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
19	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
20	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
21	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
22	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
23	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
24	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
25	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
26	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Rysunek 2.5. Tablica Johanna Trithemiusa.

Sposób szyfrowania zaproponowany przez Trithemiusa jest bardzo prosty szyfrant szyfrując pierwszą literę tekstu korzysta z alfabetu oznaczonego numerem jeden. Gdy chce zaszyfrować następny wiersz korzysta z drugiego itd. za każdym razem biorąc nowy alfabet tajny. Wydaje się, że sposób jest nie do złamania, lecz gdy kryptolog zna wyżej zastosowany algorytm to bez problemu odszyfruje wiadomość. Rozwiązaniem jest korzystanie z różnych alfabetów w różnej konfiguracji tak jak to zrobił w XVI wieku Blaise de Vigenere. Dzięki czemu uzyskał szyfr, który nie został złamany aż do wieku XIX, kiedy to Brytyjczyk Charlesa Babbage’a złamał szyfr Blaise de Vigenere. Jak tego dokonał? Otóż zauważył, że jeśli kryptolog zdobędzie informacje o długości klucza to będzie wiedział które litery są zaszyfrowane przy użyciu identycznego alfabetu tajnego. Gdy kryptolog pozna tę informację będzie mógł zaatakować te same grupy liter przy użyciu analizy częstości tak jak to robiono dla szyfrów monoalfabetycznych. Jak więc poznać długość klucza? Otóż wiemy, że zawsze co n liter zostaje powtórzony ten sam alfabet więc co n liter może istnieć sytuacja, że zostaną powtórzone te

same grupy liter przy użyciu tego samego klucza znajdując, ile dzieli poszczególne grupy liter np. 16 32 40 44 wystarczy wybrać wspólny dzielnik ww. odległości. Dzięki czemu poznamy, że długość klucza to cztery. Teraz wystarczy przyporządkować litery do siebie które zostały zaszyfrowane przy użyciu tego samego alfabetu tajnego i możemy atakować zaszyfrowany tekst za pomocą analizy częstości występowania tych słów w alfabecie języka tajnego i pogrupować je tak jak zostało to napisane kilka stron wcześniej. Kolejnym sposobem szyfrowania jest użycie tzw. **Szyfru Digraficznego**. Szyfr digraficzny jest to odmienny sposób szyfrowania, ponieważ nie skupiano się na pojedynczej literze, lecz dzielono wyraz na pary. Na przykład wyraz cztery zostałby podzielony tak: CZ TE RY. Teraz przy użyciu określonego sposobu można było szyfrować nie pojedynczą, lecz parę liter. W wyniku czego otrzymywaliśmy bardzo bezpieczny sposób utajniania informacji. W 1854 roku Charles Wheatstone'a opublikował pierwszy szyfr digraficzny. Szyfr ten opierał się o tablice o szerokości i długości 5. Wpisywano tam alfabet od początku do końca można było również dodać słowo klucz wewnątrz co zmieniało porządek liter.

P	R	Y	M	A
B	C	D	E	F
G	H	IJ	K	L
N	O	Q	S	T
U	V	W	X	Z

Rysunek 2.6. Szyfr Digraficzny Charlesa Wheatstone.

Tablica oparta o te zasady może wyglądać tak jak na powyższym rysunku. Kluczem będzie słowo *PRYMA*. Pomimo że ojcostwo szyfru przypada dla Charlesa Wheatstone to jest on przypisywany innemu uczonemu Lyon-owi Playfair. Wynika to z tego, że Charles i Lyon jak można zobaczyć na zdjęciach przedstawionych chociażby na stronach internetowych byli do siebie bardzo podobni co sprawiło, że całe zasługi przypadły właśnie Lyon-owi Playfair. Stało się tak ponieważ upowszechnił on ten sposób szyfrowania. Dlatego szyfr ten nazywa się szyfrem Playfair. Postaram się na słowie „*PRACA DYPLOMOWA*” przedstawić w jaki sposób były szyfrowane wiadomości za pomocą wyższego szyfru:

- 1) Dzielimy wyraz na pary PR AC AD YP LO MO WA

2) Teraz patrzymy, czy litery powtarzają się

a) w tej samej kolumnie

b) w tym samym wierszu

Jeśli litery powtarzają się w kolumnie to zastępujemy je literami występującymi poniżej ich. Gdy powtarzają się w wierszu to należy zamienić je biorąc litery po ich prawej stronie. W obu przypadkach mieliśmy do czynienia z cyklicznością to znaczy, że ostatnia litera np. A w wierszu pierwszym była zastępowana przez P. Co, jeśli litery nie znajdują się ani w kolumnie, ani w wierszu? Wtedy należy zaszyfrować daną literę przy użyciu tego samego wiersza, lecz która literę mamy wziąć wyznacza litera w innej kolumnie np. RE to literę R zamieniamy w M, a literę E zamieniamy w C otrzymując MC.

3) Ze słowa „PRACA DYPLMOWA” otrzymamy RE RF YF MR HT RS ZY Rerfy Fmrhtrszy.

Jak można złamać tego typu szyfr otóż możemy zauważyć, że pary takich samych liter na pewno zostają zakodowane w taki sam sposób więc należałoby nie analizować częstość liter, tylko częstość występowania par w danym języku i tak próbować odszyfrować wiadomość. W następnym podrozdziale skupię się na kolejnym ważnym elemencie nie tylko z punktu widzenia bezpiecznej transmisji, ale głównie z punktu historycznego. Uważam, że historię Enigmy powinien znać każdy Polak. Dlatego że złamanie szyfru Enigmy 31 grudnia 1932 roku miało kolosalne znaczenie nie tylko dla Polski, ale dla całego świata, ponieważ sukces ten już wtedy wpłynął na zwycięstwo w 1945 roku. Dodatkowo uważam, że pamięć o tym sukcesie jest sprawą ważną i trzeba o niej pamiętać i przypominać. Postaram się również wspomnieć, kiedy to po raz pierwszy Polski wywiad złamał szyfr obcego państwa doprowadzając dzięki temu do kolejnego ważnego zwycięstwa naszej armii, a zwycięstwo było równie ważne jak Bitwa pod Wiedniem.

2.3 Enigma.

Enigma ktoś by mógł zapytać jakie przesłanie i jaką naukę niesie w dzisiejszym cyfrowym świecie to po dokonaniu Polskiej grupy Kryptologicznej mogę wskazać na jedno, że nie algorytm szyfrujący gwarantuje bezpieczeństwo, a klucz użyty do jego działania. Gdyby za każdą nową transmisją klucz był inny to ilość materiału byłaby niewystarczająca do jego złamania. Lecz wszyscy Niemieccy matematycy i inżynierzy, którzy pracowali byli przekonani, że nie da się złamać tej maszyny. Uważano, że aby złamać ten algorytm zastosowany w Enigmie kryptolog potrzebowałby kilku tysięcy lat. Jak więc Polskim kryptologom udało się rozszyfrować tą maszynę, która uważana była za niemożliwą do rozszyfrowania. Otóż na

początku przyjrzyjmy się historii Polskiej kryptografii. Jest rok 1919 Polska jest uwikłana w konflikt jednym z największych mocarstw świata. Armia Czerwona która można powiedzieć narodziła się po rewolucji chciała opanować świat. Ta sztuka mogłaby się udać, gdyby nie jeden kraj, nad Wisłą który zatrzymał jej pochód na zachodnie państwa. W 1919 roku jeden z oficerów Wojska Polskiego porucznik Jan Kowalewski złamał szyfry Armii Czerwonej. Fakt ten był nie tylko ważny dla tamtego okresu, ale również dla początków Polskiej Kryptologii, która wtedy prawie nie istniała. Do tej pory wywiad Polski nie łamał szyfrów, a korzystał z przejętych kluczy i informacji o sposobie szyfrowania we wrogim państwie. Wtedy to po tym sukcesie Dowództwo Wojska Polskiego zdecydowało o utworzeniu Polskiej grypy Kryptologicznej którzy mieli łamać szyfry Armii Czerwonej. Tak narodził się początek sukcesu, który zaowocował w 1932 roku. To ten początek przyczynił się do największego sukcesu, który był decydujący podczas II Wojny Światowej. Państwa zachodu po Pierwszej Wojnie Światowej wiedziały jak ważną rolę pełni odczytywanie szyfrów, ponieważ takim sposobem udało się wygrać wojnę, lecz mimo dużego doświadczenia nie potrafili złamać szyfrów Enigmy. Uważali, że jest to niemożliwe do wykonania, dlatego też nie zajmowali się tym. Jednak Polacy jako naród zdolny którzy potrafią dokonać niemożliwego znów pokazali swoją klasę. Zebrali grupę najlepszych matematyków, którzy pracowali nad złamaniem Enigmy. I tak trzech Polskich matematyków Henryk Zygalski, Jerzy Różycki i Marian Rejewski mimo młodego wieku złamali kod Enigmy. Udało im się 31 grudnia 1932 roku odczytać pierwsze szyfrogramy. Następnie opracowali skuteczne sposoby łamania klucza które poznamy w dalszej części rozważań. Ten sukces zaowocował i pozwolił skrócić, jak mówią naukowcy II Wojnę Światową o 3 lata co uchroniło wiele ludzi przed ŚMIERCIA, a dokładnie około 30mln ludzi. Trzy osoby każda z nich można powiedzieć uratowała 10mln ludzi. Dzięki dokonaniu Polskich matematyków i ich ustaleniom Brytyjczycy mogli odszyfrowywać depesze podczas II Wojny Światowej. Jak więc działa i jak jest zbudowana Enigma? Głównym elementem maszyny były wirniki. Pierwszą maszynę skonstruował Amerykanin Eduard Hugo Hebern. Wynalazek Eduarda Hugo Heberna wykorzystywał dwie maszyny, do pisania które były ze sobą połączone, a dane były przesyłane za pomocą impulsu elektrycznego. Impulsy te biegły od jednej maszyny do drugiej, lecz pomiędzy nimi znajdowały się wymienione wcześniej wirniki dzięki którym możliwa była zmiana połączeń. Szyfr uzyskany w tej maszynie to zwykły szyfr opierający się na jednym alfabecie, ponieważ naciśnięcie klawisza w jednej maszynie uruchamiało klawisz w drugiej maszynie. W podobnym czasie skonstruowano maszyny, które w zasadzie działania przypominały tę Eduarda Hugo Heberna. Jeden z konstruktorów był Holendrem drugi zaś Niemcem. Jednak wynalazek, który powinien zostać

wzięty pod uwagę przez Niemieckie dowództwo to w 1918 roku kraj, który był zmęczony wojną nie miał ochoty się zbroić i walczyć, a dodatkowo musiał spełnić postanowienia jakie nałożył traktat Wersalski. Artur Scherbius Inżynier, który zaproponował swojemu krajowi musiał poczekać na użycie jego wynalazku przez swój kraj jakim były Niemcy aż do przejęcia władzy przez Adolfa Hitlera. Zmiana w polityce państwa oraz modernizacja kraju, a także armii przyczyniła się do wzrostu zainteresowania maszyną. Dzięki czemu o tym wynalazku usłyszał cały świat. A wiele osób uważało za niemożliwe odkrycie i złamanie sposobu szyfrowania tej maszyny. Wynalazek Artura Scherbiusa w zasadzie działania przypominał wynalazek amerykański. Maszyna, w której główną część stanowiły wirniki, połączenia również były oparte na wymianie impulsów elektrycznych, lecz oprócz szyfrowania umożliwiała także deszyfrowanie wiadomości. Dzięki czemu ułatwiała i przyspieszała wymianę informacji. Jednak jak każde ułatwienie ma swoją wadę. Powodowało to, że litera A nigdy nie była zaszyfrowana przez nią samą co może stanowić cenną informację przy deszyfracji. Zasada działania była prosta naciśnięcie klawisza powodowało podstawienie np. za literę A litery B która była podświetlana na drugiej klawiaturze. Po naciśnięciu klawisza nie tylko podstawiano za literę A literę B, ponieważ gdyby tylko tak to wyglądało byłoby to bardzo proste do deszyfracji. Jednak, gdy wprowadzona została litera następowało obrócenie wirnika dzięki czemu za każdym naciśnięciem ta sama litera była kodowana w inny sposób, lecz nigdy nie była zakodowana przez nią samą. Aby odczytać wiadomość na drugiej maszynie należało w taki sam sposób ustawić trzy rzeczy:

- które wirniki będą wykorzystane,
- w jakim porządku zostały ustawione wirniki,
- jakie są wyjściowe pozycje wirników służących do szyfrowania.

Zdjęcie Niemieckiej maszyny szyfrującej która przysporzyła wielu osobom nieprzespane noce przedstawia rysunek 2.7. Jak to się stało, że udało się odczytać pierwsze szyfry Polskim Kryptologom. Otóż oprócz dużych umiejętności matematycznych Henryka Zygalskiego, Jerzego Różyckiego i Marian Rejewskiego duże zasługi również należą się Polskiemu wywiadowi, który zdobył informację o tym jak są przesyłane klucze, że występują na początku wiadomości powtórzone dwukrotnie dla pewności, że zostaną właściwie odebrane. Myślę, że duplikowanie kluczy miało charakter sprawdzania wysłanych danych. Jeżeli klucz pierwszy różnił się od drugiego to by znaczyło, że wystąpił błąd podczas przesyłania wiadomości. Wtedy należałoby powtórzyć całą operację. Dodatkowo szyfranci III Rzeszy nie byli zbyt kreatywni.



Rysunek 2.7. Enigma.

Zaszyfrowane wiadomości były dosyć do siebie podobne. Zaczynały i kończyły się w takim samym sposób. Także powtarzano liczne słowa i treści jest to zrozumiałe może to był sposób w jaki badano, czy dane dotarły w prawidłowy sposób można to określić jako punkty kontrolne. Oczywiście taki sposób daje pewność przesyłanych informacji, ale zmniejsza poziom bezpieczeństwa. Ważnym elementem w walce z Enigmą było przechwycenie jej cywilnej wersji w 1927 roku. Egzemplarz ten pomógł zapoznać się z budową i zasadą działania Niemieckiej maszyny szyfrującej. Wojsko niemieckie nadal doskonaliło swoją maszynę. Sprawiało to, że Enigma była coraz trudniejsza do odszyfrowania. W odpowiedzi nasi kryptolodzy stworzyli maszynę, która pomagała w obliczeniach. Cyklometr służył do wyszukiwania pewnych podobnych permutacji. Lecz i ta maszyna zawiodła. Dlatego stworzono następną maszynę Bombę. Nazwano ją tak ponieważ odgłos jej pracy przypominał cykającą bombę zegarową. Jednym z mniej znanych urządzeń, które pomagały w walce z kodem Enigmy to płachty opracowane 1938 roku przez Henryka Zygalskiego. Po wybuchu wojny prace przeniosły się do Wielkiej Brytanii. Tam też opracowano nowy sposób analizy niemieckiego szyfru. Podczas badania kryptogramu kryptolog poddawał dokładnej analizie

otrzymaną wiadomość. Skoro wiadomo było, że żadna litera np. A nigdy nie była szyfrowana przez nią samą można było próbować odszyfrować wiadomość dopasowując domniemany wyraz do wiadomości. Otóż, jeśli spodziewano się, że w wiadomości występuje niemieckie słowo Angriff oznaczające atak można było próbować odszyfrować wiadomość poprzez szukanie miejsca, w którym nie znajduje się żadne dopasowywanie sposób ten został przedstawiony na poniższym rysunku.

A	C	W	X	Z	F	I	H	A	F
A	N	G	R	I	F	F			

A	C	W	X	Z	F	I	H	A	F
	A	N	G	R	I	F	F		

A	C	W	X	Z	F	I	H	A	F
		A	N	G	R	I	F	F	

A	C	W	X	Z	F	I	H	A	F
			A	N	G	R	I	F	F

Rysunek 2.8. Analiza kryptologiczna kodu Enigmy wykorzystująca tzw. negatywny wzorzec.

Wyniku analizy kodu Enigmy i zwiększającej się złożoności. Bomby opracowane przez Polski zespół kryptologiczny nie potrafiły sprostać tym zmianom. Dlatego też do walki z Enigmą wynaleziono nowe urządzenie. Potrafiło przechowywać dane, a nazwano je Colossus. Uważam, że można stwierdzić, że był to pierwszy komputer. Lecz prawda o nim była ukrywana przez wiele lat. Uważam, że gdyby nie prace naszych kryptologów nad Enigmą to wszelkie późniejsze dokonania nawet Colossus mógłby nie zostać tak szybko stworzony jak to miało miejsce. Ponieważ wszyscy uważali, że Enigmy nie da się złamać. Więc skoro się nie da to może nawet nikt nie chciałby dalej próbować, ale skoro Polacy dali radę więc i nam się uda osiągnąć sukces. Oczywiście nie można wszystkiego przypisywać dokonaniom Polskich matematyków, ale uważam, że pamięć o ich dokonaniach powinna być szerzona i rozprzestrzeniana tak aby nikt o nich nie zapomniał. W następnym podrozdziale skupię się na teoretycznych rozważaniach problemu dotyczącego szyfru z kluczem jednorazowym.

2.3 Problem szyfru z kluczem jednorazowym. Jak go rozwiązać?

Analizując szyfry może nasuwać się pytanie czy po tylu latach można znaleźć szyfr nie do złamania? Oczywiście że tak od ponad stu lat znany jest szyfr, którego nie da się złamać wykorzystuje on jednorazowy klucz powinien być równej bądź większej długości niż szyfrowana wiadomość. Dodatkowo klucz musi być użyty tylko raz. Nie można użyć tego samego klucza więcej niż dwa razy. Klucz użyty do szyfrowania ma być jednorazowy nie można użyć klucza drugi raz. Więc skoro istnieje 100% sposób zabezpieczenia informacji to, dlaczego jest on wykorzystywany tylko do nielicznych celów na przykład tzw. Gorąca Linia pomiędzy największymi mocarstwami USA i Rosji. Otóż podczas korzystania z tego sposobu rodzą się problemy które jest bardzo ciężko rozwiązać. Mamy trzy główne problemy:

- jak zachować dostateczną losowość klucza? Pomimo rozwoju dziedzin matematyki oraz informatyki generowanie elementów losowych jest nadal skomplikowanym zagadnieniem i nadal są poszukiwane metody jak efektywnie generować losowe ciągi liczb.

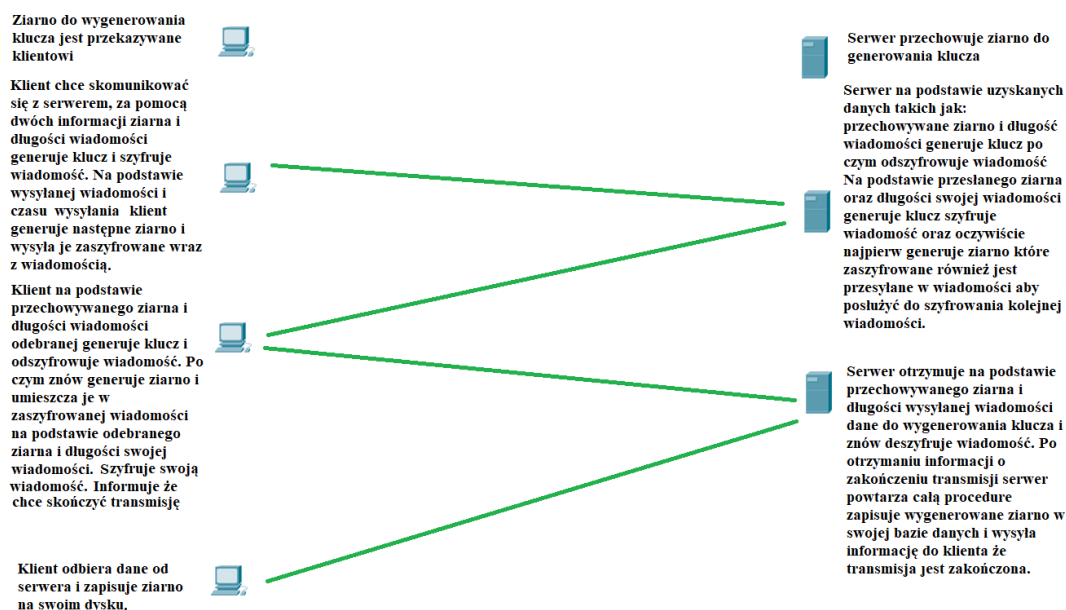
- jak za każdą nową wiadomością generować nowy jednorazowy klucz, który będzie o równej bądź większej długości niż wiadomość. Jeśli zastosujemy klucz równej długości to ilość danych zwiększy się dwukrotnie również generowanie klucza jest to rzecz, która również opóźnia cały proces.

- rodzi się pytanie w jaki sposób dostarczyć klucz do adresata, jeśli jest to transmisja pomiędzy dwoma bądź kilkoma osobami to znane są rozwiązania, że po prostu dostarczamy tablicę z kluczami do adresata. Jednak, gdy chcielibyśmy, aby ten sposób działał pomiędzy milionami osób to jest rzecz niewykonalna. Ponieważ jak bezpiecznie dostarczyć klucz i jak go zaszyfrować jakim szyfrem? Dlatego rozwiązanie te idealnie sprawdza się pomiędzy niewielką grupą użytkowników.

Lecz czy można rozwiązać te problemy czy można zaproponować jakieś rozwiązanie? Według mnie można, ponieważ nie istnieją problemy, których nie da się rozwiązać trzeba tylko dobrze szukać odpowiedzi. Problem generowania liczb losowych powinien zostać u matematyków, lecz warto byłoby szukać rozwiązania, które wykorzystuje pewne ziarno co oczywiście osłabia nasze rozwiązanie, bo poznanie ziarna skutkowałoby odszyfrowaniem wiadomości, lecz nie jest to takie proste. Uważam, że można rozwiązać ten problem oczywiście jak każde rozwiązanie ma pewne wady i zalety. Jak więc można spróbować rozwiązać powyższe problemy. Pomyślałem, że każdemu z nas najbardziej zależy na bezpiecznej wymianie informacji pomiędzy naszym bankiem, a komputerem bądź dowolnym innym urządzeniem. W moim sposobie rozwiązania problemu klucz byłby przekazywany tylko raz, a raczej ziarno do generowania klucza. Następnie każda nowa wiadomość zawierałaby ziarno do następnego

nowego klucza wraz z długość wiadomości. Ziarno generowane na podstawie czasu utworzenia wiadomości byłoby bardzo dobrym rozwiązaniem, ponieważ nie da się w tym samym czasie wysłać takich samych wiadomości dodatkowo można byłoby dodawać liczbę losową tak aby przy milionach czy nawet miliardach użytkowników nie trafił się ten sam klucz. Takie rozwiązanie jest 100% gwarancją, że wiadomości nie zostaną odszyfrowane, jeśli pierwsza wymiana jest bezpieczna wiemy, że następna również jest bezpieczna. Nawet jeśli algorytm generowania klucza oraz liczb pseudolosowych będzie znany to nie ma możliwości poznać klucza bez znajomości danych wejściowych bez znajomości całej wiadomości. Dlatego mój sposób rozwiązuje problem dystrybucji klucza dla następnych wiadomości. Następnie po zakończeniu transmisji komputer oraz baza danych lubi drugi komputer musiałby przechowywać klucz co jest na pewno dodatkowym problemem, ponieważ zabiera dodatkowe miejsce. Warto również zastosować szyfrowanie dysku. Ponieważ trzeba też zwrócić uwagę co, jeśli ktoś włamie się na komputer wtedy może wykraść ziarno. Dlatego aplikacja musiałaby szyfrować część dysku albo korzystać z zaszyfrowanego dysku przenośnego który byłby szyfrowany wcześniej. Cały sposób dystrybucji i uzgadniania klucza opisuje rysunek poniżej.

Przykładowa transmisja



Rysunek 2.9. Szyfrowanie z kluczem jednorazowym w nowy sposób.

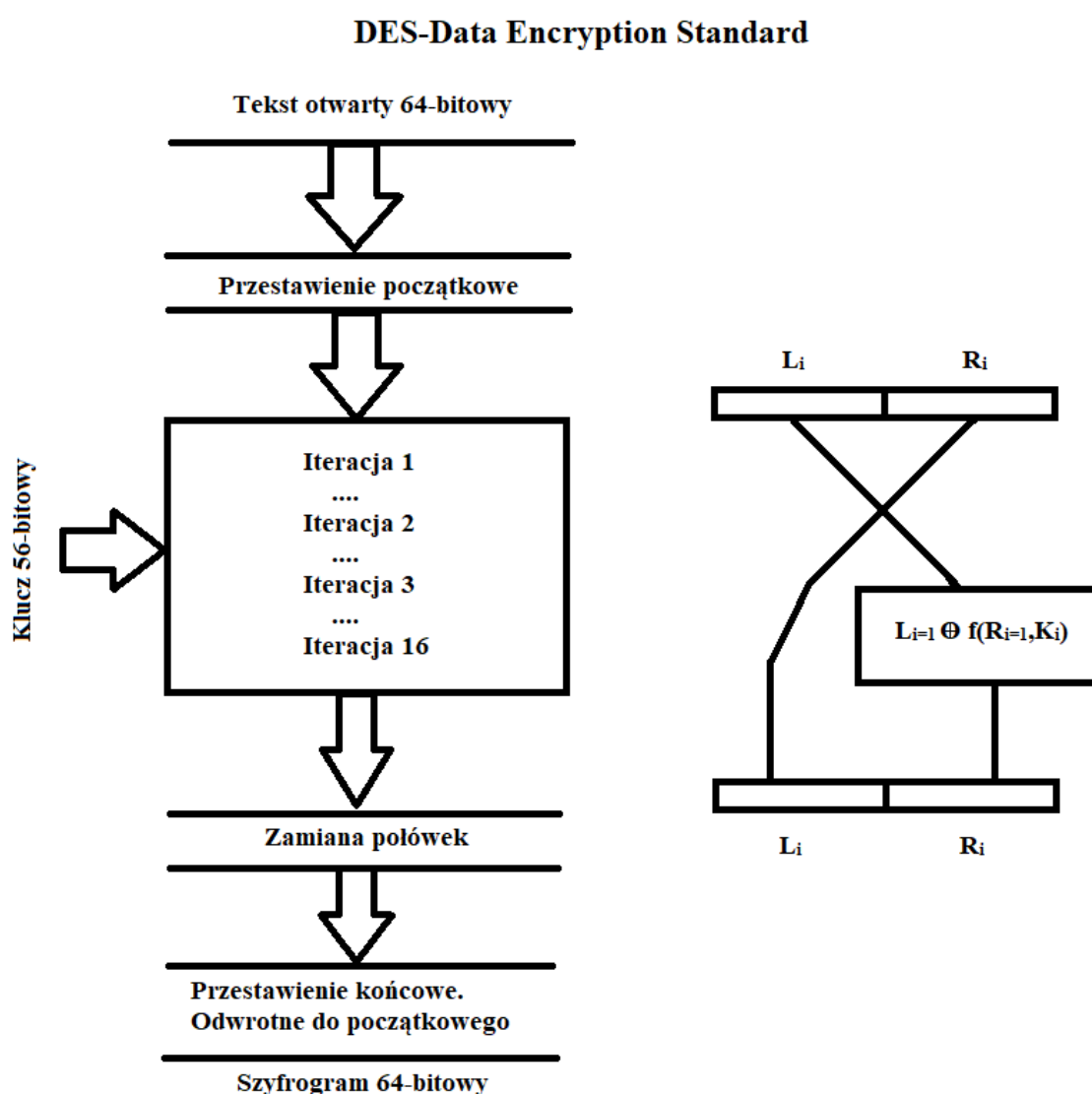
Uważam, że sposób ten rozwiązuje problem z kluczem jednorazowym. Problem pierwszy nawet jeśli kiedykolwiek klucz się powtórzy to atakujący musi o tym wiedzieć, że tak się stało. Nawet jeśli odczyta wiadomość, która ma sens to na pewno można dobrać inne klucze, dla których ta wiadomość też będzie miała logiczny sens. Więc ciężko jest się dowiedzieć czy

rzeczywiście ponownie użyto ten sam klucz. Tylko przy takim samym ziarnie i tej samej długości wiadomości uzyskujemy ten sam klucz. Problem dystrybucji który uważam za główny problem został rozwiązany. Uważam, że skoro bank przekazuje login i hasło mógłby dodatkowo przesyłać ziarno do generowania klucza. Dodatkowo klient potrzebowałby aplikacji banku która zajmowałaby się szyfrowaniem przekazanego ziarna oraz posiadałoby klawiaturę, w którą wpisujemy ziarno za pomocą kliknięć tak aby dane z klawiatury nie zostały przekazane. W następnym podrozdziale omówię algorytmy kryptograficzne takie jak AES, RSA i DES.

2.4 Algorytmy kryptograficzne w cyfrowym świecie.

Z nastaniem komputerów maszyny szyfrujące takie jak Enigma przestały pełnić tak istotną rolę jak przed istnieniem dużych maszyn obliczeniowych. Maszyny szyfrujące posiadały ograniczenia co do możliwości technicznych które były potrzebne przy budowie maszyn. Algorytmy opracowane wraz z pojawieniem się komputerów mogły być bardziej skomplikowane. Mogły opierać się na bardzo skomplikowanych obliczeniach, które komputer wykonywał w bardzo krótkim czasie. Dodatkowo komputeryzacja przyspieszyła samo szyfrowanie teraz wystarczyło tylko wpisać cały tekst i po jednym kliknięciu mogliśmy otrzymać zaszyfrowany gotowy do wysłania tekst. Uważam, że, z nadejściem ery Internetu płatności cyfrowej czy przesyłanych informacji na wiele kilometrów wymusiło szukanie sposobów, aby zabezpieczyć przesyłanie danych przed nieuprawnionym dostępem. Dzięki czemu mogliśmy swobodnie przysyłać informację wiedząc, że nikt nieuprawniony ich nie odczyta. Oczywiście rozwój komputerów jak i potężna moc jaką daje kryptografia powoduje, że rządzący starają się to ograniczać. Otóż powód jest prosty nie możemy zwykłemu człowiekowi dać możliwości do przesyłania danych, których nie możemy odczytać. Dlatego że każdy od dobrego człowieka po terrorystę mógłby bezpiecznie komunikować się z drugą osobą. Oczywiście jeśli jest to normalny człowiek, który korzysta z banku albo pisze list miłosny do drugiej osoby to wiadomo, że taka osoba nie zagraża. Natomiast jeśli dalibyśmy 100% bezpieczeństwo, że terrorysta, pedofil i wiele innych osób, które korzystają z Internetu mogą wysyłać bezkarnie różne niebezpieczne dane, a my nie mamy żadnej możliwości ataku to jesteśmy na przegranej pozycji. Związku z tym korzystanie nawet z bezpiecznej sieci nigdy nie będzie bezpieczne, ponieważ już na samym wstępie nikomu na tym nie zależy. Najbezpieczniej jest po prostu nie korzystać z sieci tylko przekazywać informacje bez otoczenia elektroniką wtedy mamy pewność, że nikt tego nie usłyszy co my myślimy. Wszystkie te rozważania powinny cofnąć nas do roku 1976 gdzie wprowadzono sposób szyfrowania DES. DES jest to sposób szyfrowania, którego autorem był Horst Feistel. W początkowej fazie projektowania opierał się na kluczu 128-bitowym, lecz poziom bezpieczeństwa był tak wysoki, że NSA nie

zgodziło się na jego upowszechnienie z kluczem 128 bitowym. NSA skróciło rozmiar bitów z 128 do 56 co dawało możliwość odszyfrowania wiadomości przez służby USA. Oczywiście nadal mieliśmy klucz, który dawał dość wysoki stopień bezpieczeństwa, lecz wiadomości nie były 100% bezpieczne, ponieważ ktoś może je odczytać. Przyjmuje się, że skrócenie liczby bitów klucza o jeden, o połowę zmniejszało bezpieczeństwo systemu DES. W związku z tym rozpoczęły się liczne protesty które nic nie wnosiły, a w roku 1976 został przyjęty nowy standard DES. Algorytm szyfrowania DES to algorytm symetryczny to znaczy, że do zaszyfrowania i odszyfrowania wykorzystuje się jeden klucz. Algorytm szyfrujący DES składający się z 19 kroków przedstawia rysunek poniżej.

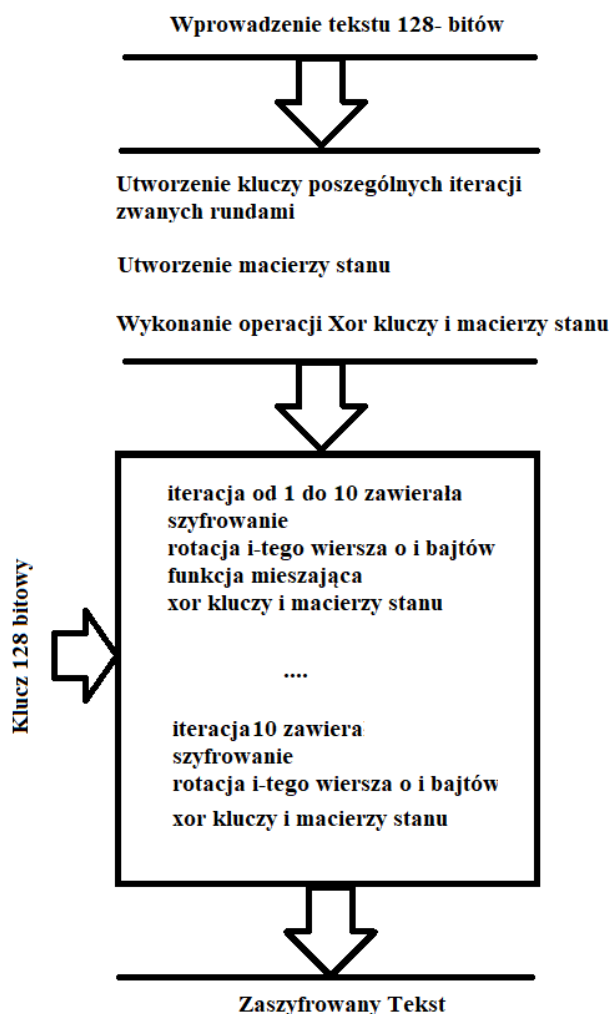


Rysunek 2.10. Algorytm szyfrowania symetrycznego DES.

Trafiający tekst jawny do algorytmu DES w pierwszym kroku został poddawany transpozycji następnie wykonywana była pętla, w której brano każdą inną część klucza, schemat tego

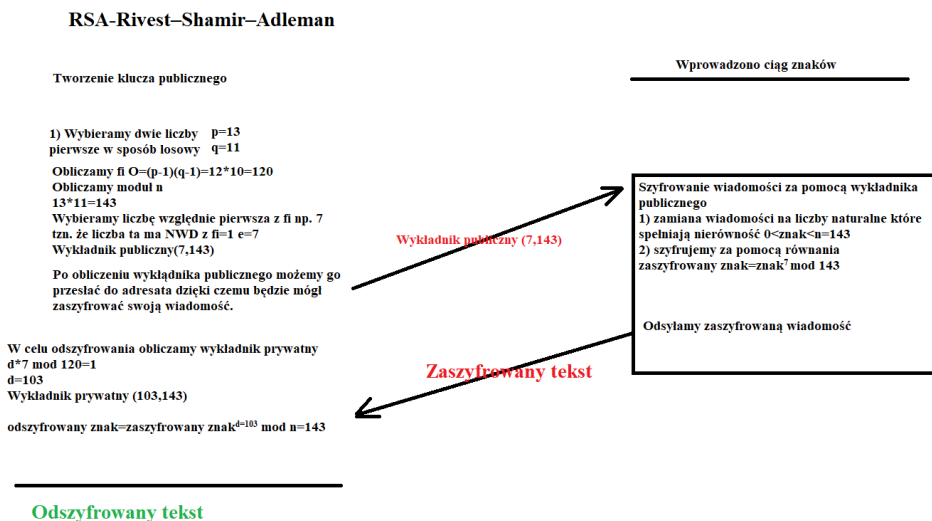
działania widać na rysunku obok. Gdy pętla się zakończyła połówki były zamieniane i na końcu w ostatnim kroku następowała permutacja tych bloków danych wejściowych. Ostatni etap przebiegał w negacji do kroku pierwszego. Tak wyglądał system DES wyparty w późniejszych latach przez AES z powodu braku dostatecznego zabezpieczenia danych. Modyfikacja Triple DES również została wyparta na rzecz AES. Triple DES to zmodernizowana wersja DES ze wsteczną kompatybilnością. Dzięki czemu płynnie nastąpiła wymiana tych wersji. Triple DES różnił się od poprzednika, lecz niezbyt dużo. Algorytm ten polegał na trzykrotnym wykorzystaniu algorytmu DES stąd wywodzi się nazwa Triple DES. Wykorzystywał również dwa klucze, za których pomocą był szyfrowany tekst. Schemat szyfrowania był prosty najpierw następuje szyfrowanie za pomocą pierwszego klucza następnie tekst był rozszyfrowywany za pomocą klucza drugiego po czym następnie szyfrowano znów kluczem pierwszym. Dzięki temu, jeśli użyliśmy jednego klucza to otrzymywaliśmy zwykły DES co pozwalało na zachowanie wstecznej kompatybilności. Klucz Triple DES miał 112 albo 168 bitów długości. Następnym algorytmem to AES który wyparł algorytmy DES i Triple DES. W odpowiedzi na różnego rodzaju głosy sprzeciwu podczas tworzenia algorytmu DES instytucja NIST ogłosiła otwarty konkurs na nowy sposób szyfrowania który miał nazywać się AES Advanced Encryption Standard. Wygrał pomysł Joana Deamena oraz Vincenta Rijmen, a algorytm nosił nazwę Rijndael od ich nazwisk. Klucze użyte do szyfrowania mają długość 128 lub 256 bitów. AES nawet przy kluczu 128 bitowym był niezwykle bezpieczny oznaczało to, że istnieje 2^{128} kluczy które trzeba sprawdzić przy odszyfrowywaniu wiadomości. Operując potężną mocą obliczeniową sprawdzenie wszystkich kluczy zajęłoby więcej czasu niż wypalenie się słońca w naszym układzie. Dlatego AES jest bezpiecznym algorytmem. Wielkość bloków danych wejściowych podwoiła się w stosunku do DES to 128 bitów. Sposób działania algorytmu pokazuje rysunek poniżej.

AES- Advanced Encryption Standard



Rysunek 2 11. Algorytm szyfrowania symetrycznego AES.

Ostatnim algorytmem szyfrującym który opiszę będzie RSA. RSA jest to sposób, który przeczy poprzednim opisanym rozwiązaniom. Według nich klucz powinien być tajny. W tym przypadku klucz był publiczny. RSA to asymetryczny algorytm co oznacza, że jeden klucz służy do szyfrowania drugi do odszyfrowywania danych. Jak to się stało, że zastosowanie klucza publicznego jest możliwe. Otóż autorzy tego sposobu Rona Rivesta, Adi Shamira i Leonarda Adlemana stworzyła algorytm, który opierał się na operacjach na dużych liczbach pierwszych. Mimo że możemy poznać klucz użyty do szyfrowania to nie jest to takie proste otóż rozwiązanie równania matematycznego służącego poznaniu klucza było niewykonalne w racjonalnym czasie. Sposób działania algorytmu RSA pokazuje poniższy rysunek.



Rysunek 2.12. Zasada działania algorytmu RSA.

Wszyscy znają klucz publiczny którym można tylko zaszyfrować, lecz nie można odczytać wiadomości. Lecz wystarczy prosty atak MiTM i sposób ten jest bardzo łatwo obchodzony. Odbieramy wykładnik publiczny, tworzymy swój wykładnik publiczny i prywatny następnie swój wykładnik publiczny wysyłamy do interesującej nas osoby dzięki czemu ona szyfruje wiadomość my ją odszyfrowujemy, czytamy zmieniamy albo i nie następnie wysyłamy ponownie szyfrując za pomocą wcześniej otrzymanego klucza publicznego.

2.5 Podsumowanie.

Od zarania dziejów człowiek próbuje zabezpieczać swoje dane. Lecz jak możemy zobaczyć nie ma idealnej metody jedna jest lepsza druga gorsza. Wszystkie natomiast mają jedną wadę są budowane przez człowieka, człowiek jest istotą, która popełnia błędy i się myli. Dlatego żaden algorytm szyfrujący nie będzie idealny zawsze będzie miał swoje wady. Pomimo tego należy szukać dobrych rozwiązań i zastanawiać się jak udoskonalić to co posiadamy. Można powiedzieć, że kryptografia to sztuka ciągłego doskonalenia. Ponieważ każdy kto tworzy nowy sposób stara się udoskonalić sztukę kryptografii na nowo. Od szyfru Cezara po szyfr Enigmy itd. każdy tworzył coś nowego. Złamanie Enigmy trwało 2 lata od 1930 roku do 1932. Gdyby Niemcy uruchomili Enigmę w 1939 roku może byłaby tak skomplikowana, że nikt by tego nie dokonał, a wojna byłaby długa i krwawa. Dlatego warto po prostu co jakiś czas zmieniać sposób szyfrowania tak aby przeciwnik jak poznał stary sposób musiał męczyć się nad nowym. Jeśli będziemy dość szybko zmieniać sposoby szyfrowania to przeciwnik za nim po zna stary sposób

będzie musiał męczyć się nad nowym sposobem itd. Gdy użyjemy dostatecznie dużo wariantów liczba ich może okazać się trudna do złamania. Tak jak to wynika z szyfru z kluczem jednorazowym. Pytanie tylko czy istnieje możliwość ciągłego zmieniania sposobów w taki sposób, aby przeciwnik nie mógł poznać żadnego? W każdym razie należy tworzyć nowe rozwiązania tak żeby nawet jeśli mają wady to, żeby nie można było ich rozszyfrować w rozsądnym czasie. Na przykład szyfr używany podczas potyczki powinien być tak mocny, aby podczas wydawania rozkazów poszczególnym pododdziałom przeciwnik ich w tym czasie nie mógł złamać ani poznać, lecz powinien być dostatecznie prosty, aby można było nauczyć rozszyfrowywać i szyfrować wiadomości nawet najbardziej opornego dowódcę. Tak samo wiadomości wysyłane do banku podczas których informujemy, że chcemy przelać pieniądze na inne konto powinny być zaszyfrowane takim szyfrem, że nikt do momentu wykonania operacji nie będzie mógł zakłócić tej transakcji. Po wykonaniu transakcji nieuczciwy człowiek może poznać kwotę oraz numer konta, na które została ta kwota przelana. Lecz nie będzie mógł ich zmienić szyfr spełni swoją rolę. Natomiast dane służące do uwierzytelnienia naszego konta w banku dobrze, aby nikt ich nie poznał w czasie w jakim cyklicznie zostają one zmienione. Dzięki czemu stare hasło do banku może okazać się nie przydatne, jeśli zostanie użyte tylko raz wtedy nawet jeśli ktoś pozna nasze hasło nie będzie mógł go wykorzystać, a nasze konto pozostaje bezpieczne. Uważam, że każdy szyfr w pewnym miejscu i czasie jest doskonały ważne, aby spełnił swoją rolę oraz najważniejsze, należy odpowiednio dobierać miejsca do ich użycia oraz poziom skomplikowania.

3 Program do zbierania informacji o trasie pakietów.

Wyniku mojej pracy oraz analizy pakietów powstał program, który wykorzystuje zebraną wiedzę nie tylko tą podczas pisania pracy, ale również moich studiów. Powstał program, który można powiedzieć jest tym o czym marzyłem. Nie jest obszerny nie ma miliona funkcjonalności, lecz jest owocem pracy i nauki. Jest to program, który wykorzystuję zarówno wiedzę sieciową jak również programistyczną którą zbierałem od początku nauki na studiach. Zasada jego działania opiera się na wcześniej wspomnianym protokole ICMP. Za jego pomocą badam zbieram informację o trasie pakietów. Program umożliwia zebranie informacji o urządzeniach sieciowych które występują na drodze do naszego hosta docelowego którego podaliśmy w oknie aplikacji. Program działa bardzo dobrze oraz jest przyjazny dla użytkownika. Posiada obsługę błędów oraz wykorzystuje nowoczesne podejście w programowaniu. Podczas pisania funkcji kierowałem się znaną zasadą, jeśli funkcja zajmuje więcej niż ekran komputera to oznacza, że jest zbyt duża. Każda z funkcji odpowiada jednemu zadaniu tak samo jak klasa. Nazwy funkcji są czytelne i zrozumiałe dzięki czemu nie potrzeba

używać komentarzy, ponieważ cały kod jest zrozumiały, a sama nazwa funkcji odzwierciedla jej funkcjonalność. Dzięki temu opis kodu, który zostanie przedstawiony poniżej będzie zwięzły i krótki, ponieważ funkcja sama w sobie mówi co wykonuje. Poniżej możemy zobaczyć funkcję, która realizuje najważniejsze zadanie podczas działania programu.

```
1 public sledzeniePakietow wyslijPing(sledzeniePakietow sledzenie)
2 {
3     Ping ping = new Ping();
4     try
5     {
6         PingOptions pingOptions = new PingOptions( sledzenie.TTL, true);
7
8         PingReply odpowiedz = ping.Send(sledzenie.adress,sledzenie.czasOczekiwania,sledzenie.wielkoscBuforu,pingOptions);
9         try
10        {
11            sledzenie.IPhosta = odpowiedz.Address.ToString();
12        }
13        catch
14        {
15            sledzenie.IPhosta = "000.000.000.000";
16        }
17        if (odpowiedz.Status==IPStatus.TtlExpired)
18        {
19            sledzenie.odpowiedz = " Ip hosta: " + odpowiedz.Address + " ";
20            return sledzenie;
21        }
22        if (odpowiedz.Status == IPStatus.Success)
23        {
24            sledzenie.odpowiedz = " IP hosta: "+odpowiedz.Address+" ";
25            sledzenie.zakonczPing = true;
26            return sledzenie;
27        }
28        if(odpowiedz.Status == IPStatus.TimedOut)
29        {
30            sledzenie.odpowiedz = " Upłynął limit czasu żądania: " + odpowiedz.Address + " ";
31            return sledzenie;
32        }
33    }
34    catch(Exception ex)
35    {
36        sledzenie.blad = "Błąd : "+ex.Message;
37        sledzenie.czyNastapilBlad = true;
38        return sledzenie;
39    }
40    return sledzenie;
41 }
```

Rysunek 3.1. Funkcja wysyłająca pakiet ICMP.

Funkcja wyslijPing przyjmuje jeden parametr, który jest wcześniej stworzonym obiektem. Obiekt jest to byt, który posiada pewne cechy przechowuje odpowiednie wartości, natomiast klasa jest to definicja obiektu. Klasa nazywa się po prostu sledzeniePakietow, a obiekt sledzenie od nazwy klasy. Obiekt ten zawiera cztery parametry które są automatycznie przypisywane za pomocą konstruktora podczas jego tworzenia. Konstruktor to specjalna funkcja, która ma nazwę taką samą jak klasa. Również w tej klasie na poniższym rysunku zauważymy przeciążenie konstruktora to znaczy, że dwie funkcje są tak samo nazwane, lecz wykorzystują inne parametry do swej pracy.

```

1      sledzeniePakietow()
2      {
3          this.TTL = 1;
4          this.wielkoscBuforu = Encoding.ASCII.GetBytes("aaaaaaaaaaaaaaaaaaaaaaaaaaaa");
5          this.czasOczekiwania = 120;
6          this.adress = "127.0.0.1";
7          IPhosta = "000.000.000.000";
8      }
9      public void zmienTTL(int ttl)
10     {
11         this.TTL = ttl;
12     }
13     public sledzeniePakietow(int TTL,int czasOczekiwania, byte[] wielkoscBuforu,string adress)
14     {
15         this.TTL = TTL;
16         this.wielkoscBuforu = wielkoscBuforu;
17         this.czasOczekiwania = czasOczekiwania;
18         this.adress = adress;
19         IPhosta = "000.000.000.000";
20     }

```

Rysunek 3.2. Przeciężenie.

Działanie tej funkcji jest bardzo proste po przyjęciu obiektu przechowującego wprowadzone parametry w oknie programu w 3 wersji jest tworzony obiekt Ping. Obiekt ten realizuje bardzo ważne zadanie. Wysyła pakiet ICMP do hosta o wskazanym adresie. Funkcja ta realizuje jedną funkcjonalność, lecz wykorzystując jej działanie i wiedzę zebraną podczas pisania pracy inżynierskiej możemy napisać program, który będzie wykorzystywał tę funkcję do realizacji naszego zadania. Następnie jest utworzony kolejny obiekt, który służy do ustalenia liczby skoków pakietu ICMP tzw. TTL, czyli czas życia tego pakietu. Czas życia jest to liczba, która jest pomniejszana o jeden przy przejściu przez każde urządzenie sieciowe. Ta funkcjonalność zapobiega zapętlaniu i wiecznemu krążeniu pakietów. Gdy pakiet osiągnie TTL równy 0 jest wysyłany komunikat ICMP który o tym informuje. Za przechwycenie tej cechy odpowiada wers od 17 do 21 który, gdy to nastąpi przechwytuje i zwraca nam odpowiedź. Wykorzystując zasadę działania tego mechanizmu mogłem zaprojektować program, który będzie wysyłał pakiety ICMP o kolejnych wartościach TTL co powoduje, że możemy odczytać wszystkie urządzenia, które odpowiadają na zadany komunikat protokołu ICMP. Gdy nasz komunikat osiągnie docelowy cel cała operacja jest przerywana, a naszym oczom ukazuje się całą informację, którą zebraliśmy dzięki czemu możemy poznać wszystkie urządzenia na trasie pakietu. Jest to bardzo ważne, ponieważ dzięki temu możemy wychwycić ingerencję w trasę co jest szczególnie ważne podczas wykrywania ataku MiTM, czyli z człowiekiem pośrodku który został opisany w mojej pracy inżynierskiej. Cała operacja, która polega na wywołaniu

naszej funkcji jest zawarta w klasie głównej programu, a funkcję możemy zobaczyć na poniższym rysunku.

```
1 void sledzTraseFunkcja()
2 {
3     wynik.Text = " ";
4     try
5     {
6
7         byte[] bufor;
8         string buforDane = "a";
9         for (int i = 0; i < Convert.ToInt32(rozmiarBuforu.SelectedItem) - 1; i++)
10        {
11            buforDane += "a";
12
13        }
14        bufor = Encoding.ASCII.GetBytes(buforDane);
15        sledzenie = new sledzeniePaketow(Convert.ToInt32(ttl.Value), Convert.ToInt32(czasOczekiwania.Value), bufor, ipPodane.Text.ToString());
16        sledzenie1 = new sledzeniePaketow(64, Convert.ToInt32(czasOczekiwania.Value), bufor, ipPodane.Text.ToString());
17        sledzenie1 = sledzenie1.wyslijPing(sledzenie1);
18        wynik.Text += "Rozpoczynam śledzenie trasy pakietu do " + ipPodane.Text + sledzenie1.odpowiedz + Environment.NewLine;
19
20        for (int i = 1; i <= Convert.ToInt32(ttl.Value); i++)
21        {
22            sledzenie.zmienTTL(i);
23            sledzenie = sledzenie.wyslijPing(sledzenie);
24            wynik.Text += (" TTL " + i + sledzenie.odpowiedz + Environment.NewLine).ToString();
25
26        }
27
28        if (sledzenie.czyNastapilBlad)
29            throw new Exception();
30
31        if (sledzenie.IPhosta.CompareTo(sledzenie1.IPhosta) == 0)
32        {
33            break;
34        }
35    }
36
37    }
38    catch (Exception ex)
39    {
40        MessageBox.Show("Nastąpił nieoczekiwany błąd " + ex.Message);
41    }
42 }
```

Rysunek 3.3. Najważniejsza funkcja programu.

Sama funkcja jest bardzo prosta i odpowiada tylko za jedną rzecz za wysyłanie pakietów ICMP z kolejnymi wartościami TTL począwszy od 1 do liczby, która zakończy działanie tejże funkcji. Do wykorzystania tego użyłem pętli for która spełnia te zadanie w stu procentach dodatkowo podczas wywoływania pętli używam dwóch instrukcji warunkowych do ciągłego monitorowania czy nie wystąpił błąd oraz czy nie znaleziono pakietu końcowego. Po znalezieniu jest wywoływany break który przerywa instrukcje. Instrukcje warunkową w języku C# możemy zapisać w dwóch postaciach, jeśli jest ona jednolinijkowa. Pierwszy sposób bez wykorzystania klamry, a drugi z klamrami. Całość jest dodatkowo zabezpieczona przed nieznany błądem, którego nie można przewidzieć za pomocą obsługi błędów w tym języku. Gdy nastąpi nieoczekiwany błąd w polu catch który przechwyci ten błąd również zostanie wyświetlony komunikat z błędem co może ułatwić rozpoznanie co zostało wykonane nie tak podczas użytkowania aplikacji. W następnym podrozdziale opiszę aplikację od zewnątrz i zapoznam ze sposobem jej użytkowania który jest bardzo przejrzysty.

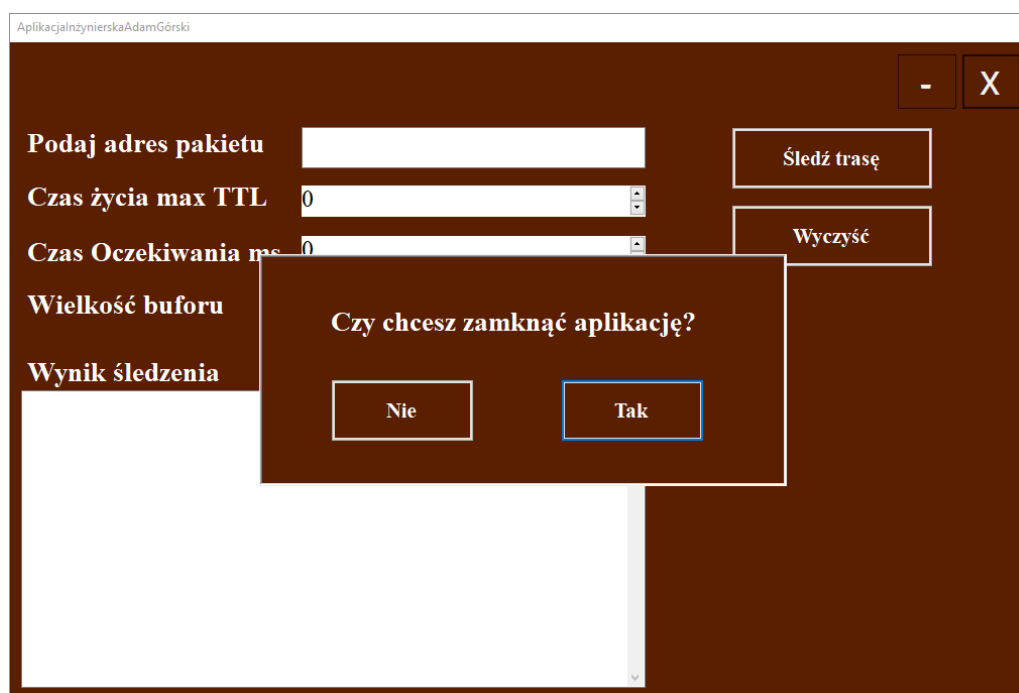
4 Opis działania programu.



Rysunek 4.1. Okno główne aplikacji.

Po zainstalowaniu i uruchomieniu programu ukaże nam się okno menu głównego. Które jest sercem całej aplikacji. Aplikacja do swej pracy potrzebuje podania czterech parametrów, adresu, czasu maksymalnego życia pakietu, który jest po prostu informacją do jakiej ilości hostów mamy szukać, czasu oczekiwania na odpowiedź ze strony aplikacji oraz wielkości buforu. Gdy posiadamy tę informację możemy zacząć pracę z naszą aplikacją. Zwrócę uwagę również, że przyciski zamykania oraz minimalizacji zaprojektowałem i oprogramowałem samodzielnie tak żeby nie korzystać z gotowego rozwiązania, ponieważ uważam, że tak jest bardziej profesjonalnie i elegancko. Wynik naszej pracy zawarty jest w dużym polu tekstowym które, gdy tylko przekroczymy ilość danych to automatycznie pojawi się pasek, dzięki któremu możemy swobodnie przesuwając wyświetlone dane. Aplikacja działa szybko w mgnieniu oka znajduje trasę pakietu dzięki czemu nie trzeba czekać i denerwować się na to aż aplikacja znajdzie drogę do wyznaczonego celu. Dodałem również prosty mechanizm, który ułatwia korzystanie, a mianowicie potwierdzenie zamykania aplikacji, które bardzo ułatwia pracę i broni przed niekontrolowanym opuszczeniem z naszej aplikacji. Jest to standard, aby potwierdzać zmiany. Zaprojektowanie funkcjonalnego okienka mniej niż pięć minut, a jest bardzo pomocne podczas korzystania z całej aplikacji. Dodatkowo mamy do dyspozycji przycisk Śledź trasę który po naciśnięciu odpala cały program oraz Wyczyść który szybko i

automatycznie czyści wszystkie dane tak aby można było szybko podjąć pracę z nowym adresem.



Rysunek 4.2. Interaktywne zamykanie aplikacji.

W mojej pracy umieściłem również dwa obrazy, które ukazują działanie całej aplikacji oraz wynik. Możemy dokładnie poznać, ile urządzeń sieciowych dzieli nas od badanej strony bądź adresu. Jest to niezmiernie ważne dla udokumentowania całej pracy. Nawet jeśli zaznaczymy za dużą trasę program przerwie w ustalonym miejscu tak aby nie powielić informacji. Dzięki temu aplikacja działa bardzo sprawnie. Wynik aplikacji przedstawiłem dla przykładowego adresu jakim jest www.helion.pl oraz 8.8.8.8 który jest adresem publicznego serwera DNS firmy Google.

AplikacjaInzynierskaAdamGórski

Podaj adres pakietu

Czas życia max TTL

Czas Oczekiwania ms

Wielkość buforu

Wynik śledzenia

```

Rozpocznam śledzenie trasy pakietu do www.helion.pl IP hosta:
188.117.147.116
TTL 1 Ip hosta: 192.168.8.1
TTL 2 Ip hosta: 172.18.123.201
TTL 3 Ip hosta: 172.18.216.67
TTL 4 Ip hosta: 172.18.216.33
TTL 5 Upłynął limit czasu żądania:
TTL 6 Ip hosta: 80.50.105.1
TTL 7 Ip hosta: 195.117.0.154
TTL 8 Ip hosta: 80.50.83.230
TTL 9 IP hosta: 188.117.147.116
  
```

Rysunek 4.3. Wynik działania aplikacji dla nazwy domenowej.

AplikacjaInzynierskaAdamGórski

Podaj adres pakietu

Czas życia max TTL

Czas Oczekiwania ms

Wielkość buforu

Wynik śledzenia

```

Rozpocznam śledzenie trasy pakietu do 8.8.8.8 IP hosta: 8.8.8.8
TTL 1 Ip hosta: 192.168.8.1
TTL 2 Ip hosta: 172.18.123.201
TTL 3 Ip hosta: 172.18.216.67
TTL 4 Ip hosta: 172.18.216.33
TTL 5 Upłynął limit czasu żądania:
TTL 6 Ip hosta: 80.50.105.129
TTL 7 Ip hosta: 195.117.0.61
TTL 8 Ip hosta: 72.14.214.158
TTL 9 Ip hosta: 108.170.228.74
TTL 10 Ip hosta: 74.125.251.103
TTL 11 IP hosta: 8.8.8.8
  
```

Rysunek 4.4. Wynik działania dla adresu IP.

Aplikacja posiada sprawdzanie informacji wprowadzonych przez użytkownika oraz zabezpieczona jest przed wyjątkami/błędami które mogłyby przerwać pracę z aplikacją. Komunikaty są czytelne, dlatego jeśli zapomnimy podać wartość aplikacja nam o tym przypomni przed wykonaniem operacji. Na poniższych rysunkach możemy zaobserwować wynik działania aplikacji. Na końcu mojej pracy umieszczę podsumowanie, które jest istotne, ponieważ wskazuje na ważny aspekt nauki i zbierania informacji bez którego ta aplikacja nie powstałaby.

Podsumowanie.

Powyższa praca to spełnienie moich marzeń i zainteresowań. W wyniku pisania pracy zebrałem niezbędną wiedzę, dzięki której mogłem napisać program przedstawiony powyżej. Dodatkowo poddałem analizie ruch sieciowy i protokoły co jest ważnym aspektem dzięki czemu dowiedziałem się o licznych lukach, na które trzeba zwrócić uwagę, aby transmisja bitowa była bezpieczna. Kolejnym ważnym osiągnięciem to zapoznanie się z dziedziną szyfrowania i zabezpieczania informacji co jest istotne podczas bezpiecznej transmisji w zbiorze hostów udostępniających swoje dane na tle innych hostów. Uświadomiłem sobie, że dobrze dobrany szyfr pozwala zabezpieczyć informację w taki sposób, aby odczytanie było niemożliwe nawet po długiej analizie w rozsądnym czasie. Zbadałem również temat szyfru z kluczem jednorazowym i wyniku analizy podałem teoretyczne rozwiązanie tego problemu. Dzięki napisaniu tej pracy osiągnąłem niezbędną wiedzę, aby móc zabezpieczyć sieć oraz zadbać o bezpieczną transmisję sieciową.

Summary.

The Engineering work is compliance my dream and interest. Consequently writing the work I collect knowledge which allowed write my application. Additionally I analyzed network traffic and protocols. It result my work I collect know about advantage and disadvantage the most important network procotols. The next important achievement it was collect neccesery knowledge about Encryption and Encoding Data. I also analyzed cipher issue One-Time Pad and propose sample resolve this problem. In result I gain knowledge neccessery to security network traffic in hosts using network media to publish data.

Spis Rysunków

Rysunek 1.1 Stos protokołów TCP/IP w formie klepsydry.	7
Rysunek 1.2. Budowa nagłówka IPv4.	9
Rysunek 1.3. Podział adresów IPv4.	10
Rysunek 1.4 Nagłówek protokołu IPv6.	11
Rysunek 1.5 Najważniejsze protokoły z podziałem na poszczególne warstwy.....	12
Rysunek 1.6. Kody odpowiedzi http.	13
Rysunek 1.7. Uścisk dłoni TCP.	14
Rysunek 1.8 Budowa datagramu ICMP.	15
Rysunek 1.9. Budowa nagłówka ARP.	16
Rysunek 1.10 Format ramki Ethernet II.	16
Rysunek 1.11. Bierne przechwytywanie ruchu sieciowego.	17
Rysunek 1.12 Okno Wireshark.	18
Rysunek 1.13. Filtr Wireshark.	19
Rysunek 1.14. Podstawowe filtry w narzędziu Wireshark.....	19
Rysunek 1.15. Najważniejsze komendy programu TCPDump.....	20
Rysunek 1.16. Analiza budowy pakietu TCPDump.	21
Rysunek 1.17. Program do czytania procesów sieciowych.....	22
Rysunek 1.18. Polecenie Netstat.	22
Rysunek 1.19. Polecenie Tasklist.....	23
Rysunek 1.20 Aktywne przechwytywanie ruchu sieciowego.	24
Rysunek 1.21 Plik hosts.	26
Rysunek 1.22. Polecenie do czyszczenia pamięci DNS.....	26
Rysunek 1.23. Serwer pośredniczący z przekierowywaniem portów.	27
Rysunek 1.24. Żądanie ARP.	29
Rysunek 1.25. Odpowiedź ARP.....	29
Rysunek 1.26. Tablica ARP przed infekcją.	31
Rysunek 1.27. Zainfekowana tablica ARP.....	32
Rysunek 2.1. Tablica Polibiusza.	36
Rysunek 2.2 Szyfr Cezara.	37
Rysunek 2.3. Szyfr przestawieniowy.	39
Rysunek 2.4. Tarcza Albertiego.	41
Rysunek 2.5. Tablica Johanna Trithemiusa.	42

Rysunek 2.6. Szyfr Digraficzny Charlesa Wheatstone.	43
Rysunek 2.7. Enigma.	47
Rysunek 2.8. Analiza kryptologiczna kodu Enigmy wykorzystująca tzw. negatywny wzorzec.	48
Rysunek 2.9. Szyfrowanie z kluczem jednorazowym w nowy sposób.....	50
Rysunek 2.10. Algorytm szyfrowania symetrycznego DES.	52
Rysunek 2.11. Algorytm szyfrowania symetrycznego AES.	54
Rysunek 2.12. Zasada działania algorytmu RSA.	55
Rysunek 3.1. Funkcja wysyłająca pakiet ICMP.....	57
Rysunek 3.2. Przeciążenie.....	58
Rysunek 3.3. Najważniejsza funkcja programu.	59
Rysunek 4.1. Okno główne aplikacji.	60
Rysunek 4.2. Interaktywne zamykanie aplikacji.....	61
Rysunek 4.3. Wynik działania aplikacji dla nazwy domenowej.....	62
Rysunek 4.4. Wynik działania dla adresu IP.....	62

Bibliografia

Banks, R. W. (2018). *Sieci komputerowe najczęstsze problemy i ich rozwiązania*. Gliwice: helion.

Barbara Halska, P. B. (2014). *Projektowanie sieci komputerowych i administrowanie sieciami*. Gliwice: helion.

Bos, A. S. (2015). *Systemy Operacyjne*. Gliwice: helion.

Forshaw, J. (2018). *Atak na sieć okiem hakera*. Gliwice: helion.

Galvin, A. S. (1993). *Podstawy Systemów Operacyjnych*. Warszawa: WNT.

Japikse, A. T. (2019). *Język C# 6.0 i Platforma .Net 4.6*. Warszawa: PWN.

Karbowski, M. (2014). *Podstawy Kryptografii*. Gliwice: helion.

Lis, M. (2016). *Praktyczny kurs C#*. Gliwice: helion.

Orłowski, S. (2007). *C# Tworzenie aplikacji sieciowych*. Gliwice: helion.

Rob Scrimger, P. L. (2002). *TCP/IP. Biblia*. Gliwice: helion.

Sosinsky, B. (2011). *Sieci komputerowe biblia*. Gliwice: helion.

Wetherall, A. T. (2012). *Sieci Komputerowe*. Gliwice: helion.

Źródła Internetowe:

Internet Engineering Task Force. (1981, September). *INTERNET CONTROL MESSAGE PROTOCOL*. Pobrano z lokalizacji: <https://datatracker.ietf.org/doc/html/rfc792>

Internet Engineering Task Force.. (1981, September). *Internet Protocol* . Pobrano z lokalizacji: <https://datatracker.ietf.org/doc/html/rfc791>

Internet Engineering Task Force.. (1981, September). *Address Resolution Protocol -- or -- Converting Network Protocol Addresses* . Pobrano z lokalizacji <https://datatracker.ietf.org/doc/html/rfc826>

Internet Engineering Task Force. (1987, November). *DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION* . Pobrano z lokalizacji <https://datatracker.ietf.org/doc/html/rfc1035>

Internet Engineering Task Force. (1987, November). *DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION* . Pobrano z lokalizacji <https://datatracker.ietf.org/doc/html/rfc1035>

Internet Engineering Task Force. (1981, September). *TRANSMISSION CONTROL PROTOCOL* . Pobrano z lokalizacji <https://datatracker.ietf.org/doc/html/rfc793>

Internet Engineering Task Force. (1980, August, 28). *User Datagram Protocol* . Pobrano z lokalizacji <https://datatracker.ietf.org/doc/html/rfc793>

