|  |
| --- |
| Warszawa,20.01.2010  Mateusz Drzymała  Robert Strulak |
| Steganograficzny komunikator w sieci LAN oparty o metodę PadSteg. |
|  |
|  |
|  |
|  |

|  |
| --- |
|  |

Spis treści

[1. Opis systemu PadSteg 2](#_Toc283461211)

[2. Decyzje projektowe 3](#_Toc283461212)

[2.1. Wybór technologii 3](#_Toc283461213)

[2.2. Najważniejsze metody biblioteki JNetPcap 3](#_Toc283461214)

[2.2.1. Pobranie listy interfejsów 3](#_Toc283461215)

[2.2.2. Przechwytywanie pakietów 3](#_Toc283461216)

[2.2.3. Ustawienie filtru 4](#_Toc283461217)

[2.2.4. Wysyłanie pakietu 4](#_Toc283461218)

[2.3. Schemat komunikacji z wykorzystaniem zbudowanej aplikacji 4](#_Toc283461219)

[2.4. Statystyka generowanego ruchu 6](#_Toc283461220)

[3. Opis aplikacji 7](#_Toc283461221)

[4. Testy 11](#_Toc283461222)

[5. Podsumowanie 12](#_Toc283461223)

[6. Bibliografia 12](#_Toc283461224)

# Opis systemu PadSteg

Steganografia jest nauką, której celem jest przekazywanie informacji w taki sposób aby sam fakt wymiany informacji był ukryty. W przeciwieństwie do kryptografii nauka ta nie koncentruję się na szyfrowaniu wiadomości jednak na zapewnieniu poufności poprzez niejawną komunikację.

Spektrum doboru nośników steganograficznych jest ogromne. Mogą to być obrazy, dźwięki, pliki tekstowe czy nawet głowy niewolników(tatuowano informację na zgolonej głowie, następnie czekano aż włosy odrosną i wysyłano dany „nośnik” do odbiorcy). Ostatnia metoda jednak dawno już wyszła z użycia. W prezentowany przez nas systemie skupiamy się na tzw. Steganografi sieciowej, która jako nośnik informacji używa protokołów sieciowych.

System PadSteg jest przykładem steganografii sieciowej międzyprotokołowej. Jego działanie oparte jest o błędne dopełnianie ramek Ethernetowych. W sieciach LAN minimalna długość ramki wynosi 64B, stąd też każda krótsza ramka musi zostać dopełniona. Początkowo ustalono, że brakujące bajty będą zastępowane zerami, jednak jak się okazało, w zależności od producenta karty sieciowej czy jej sterownika dopełnienie to często zawiera różne znaki w szczególności części pamięci jądra systemu operacyjnego. Błędy związane z dopełnianiem wymienionych powyżej ramek, określane jako zjawisko *Etherleak*  stwarza pewne możliwości, które może zostać użyte przez steganografie międzyprotokołową.

Działanie systemu PadSteg oparte jest o dopełnianie ramek Ethernetowych. W celu ustanowienia komunikacji należy wykonać następujące kroki:

1. **Inicjalizacja ukrytych węzłów – węzeł, który chce rozpocząć komunikację rozsyła wiadomość ARP-Request . W dopełnienie ramki tej wiadomości znajdują się informację, które pozwalają innym węzłom dowiedzieć się o istnieniu rozsyłającego węzła. Dopełnienie zawiera:**
2. **Losową liczbę RD,**
3. **Wynik funkcji skrótu obliczanej na podstawie wartości RD, adresu MAC nadawcy oraz identyfikatora protokołu – nośnika PID.**

**Następnie, każdy węzeł analizuje zawartość odebranej ramki, obliczana jest ta sama funkcja skrótu. Jeżeli wynik jest zgodny, to węzły są gotowe do rozpoczęcia komunikacji.**

1. **Wymiana danych – po ustalenie protokołu po którym będzie następowała komunikacja, wymiana danych może zostać rozpoczęta. Przykładowo jeżeli ustalono protokół TCP, ukryte informację mogą być umieszczane w dopełnieniach wiadomości ACK wysyłanych np. podczas transferu plików między węzłami.**

# Decyzje projektowe

## Wybór technologii

W celu zapewnienia niskopoziomowego dostępu do ramek sieciowych skorzystano z biblioteki JNetPcap napisanej w języku Java, która jest tzw „wrapperem” bibliotek Libpcap/Winpcap napisanych w językach C/C++, na których opiera się popularny sniffer sieciowy Wireshark. Jej metody umożliwiają dostęp do bufora ze wszystkimi bajtami każdej przechwyconej ramki. Dzięki temu jesteśmy wstanie zgodnie z wymaganiami projektowymi modelować przechwycone pakiety i wysyłać ukryte dane.

## Najważniejsze metody biblioteki JNetPcap

Oto spis najważniejszych metod bliblioteki JNetPcap, dzięki którym możliwe było zrealizowanie naszego projektu:

### Pobranie listy interfejsów

List<PcapIf> alldevs = **new** ArrayList<PcapIf>();

StringBuilder errbuf = **new** StringBuilder();

**int** r = Pcap.*findAllDevs*(alldevs, errbuf);

**if** (r == Pcap.*NOT\_OK* || alldevs.isEmpty()) {

System.*err*.printf("Can't read list of devices, error is %s",errbuf.toString());

**return**;

}

Metoda ta zapewnia pobranie listy interfejsów sieciowych i uzyskanie informacji na temat tych interfejsów ( adres Mac, Ip..)

### Przechwytywanie pakietów

StringBuilder errbuf = **new** StringBuilder();

Pcap pcap = Pcap.openOffline("tests/test-afs.pcap", errbuf);

PcapPacketHandler<String> handler = **new** PcapPacketHandler<String>() {

**public** **void** nextPacket(PcapPacket packet, String user) {

System.out.println("size of packet is=" + packet.size());

  }

}

pcap.loop(10, handler, "jNetPcap rocks!");

pcap.close();

Oto przykład pętli która przechwytuje pakiety i każdorazowo przy pobraniu pakietu z sieci wywołuje metodę handler.nextPacket( PcapPacket packet, String user). Dzięki temu możemy pobierać informacje na temat każdego przechwyconego pakietu i nim manipulować.

### Ustawienie filtru

PcapBpfProgram program = **new** PcapBpfProgram();

String expression = "ether proto \\arp";

**int** optimize = 0; // 0 = false

**int** netmask = Conversion.*netmask*(NetInterface.*getDevice*().getAddresses().get(0).getNetmask().getData());

**if** (pcap.compile(program, expression, optimize, netmask) != Pcap.*OK*) {

System.*err*.println(pcap.getErr());

**return**;

}

**if** (pcap.setFilter(program) != Pcap.*OK*) {

System.*err*.println(pcap.getErr());

**return**;

}

Dzieki filitrowi określonemu przez wyrażenie „expression” mamy możliwość przechwytywania pakietów które spełniają wymagany przez nas warunek. Powyżej zamieszczony jest filtr który umożliwia przechwycenie pakietów z nagłówkiem ARP.

### Wysyłanie pakietu

**if**(Pcap.*isSendPacketSupported*()){

*pcap*.sendPacket(packet.getByteArray(0, packet.size()));

}

**else** **if**(Pcap.*isInjectSupported*()){

*pcap*.inject(packet.getByteArray(0, packet.size()),0,packet.size());

}

Dzieki tym metodą jesteśmy wstanie wysłać zmodyfikowany przez nas pakiet.

## Schemat komunikacji z wykorzystaniem zbudowanej aplikacji

**Faza I – Inicjalizacja ukrytych węzłów**

Pierwszym etapem komunikacji między ukrytymi węzłami, korzystającymi z komunikatora PadSteg jest rozesłanie informacji o swojej dostępności. Jest to realizowane w następujący sposób:

* 1. W momencie włączenia aplikacji ukryty węzeł rozsyła wiadomość

ARP-Request na adres rozgłoszeniowy. Wiadomość ta zawiera niezerowe

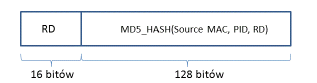
dopełnienie ramki Ethernet składające się z losowej liczby (RD) oraz wyniku działania funkcji skrótu wyliczonego na podstawie adresu MAC nadawcy, zawartości pola RD oraz wartości Identyfikatora Protokołu\Dostępności-nośnika (PID). W polu PID w fazie I znajduję się informacja o dostępności węzła.

* 1. Inny ukryty węzeł, który ma włączoną aplikacje w momencie odebrania wiadomości ARP-Request o dostępności wysyła wiadomość ARP z inną wartością PID potwierdzającą otrzymanie wiadomości o dostępności tego węzła i jednocześnie potwierdza, iż jest gotowy do przeprowadzenie rozmowy.

Oto tabela z przypisanymi wartościami identyfikatora PID:

|  |  |  |
| --- | --- | --- |
| **Protokół** | **PID** | **Znaczenie** |
| TCP | 1 | komunikacja przy użyciu protokołu TCP |
| ICMP | 2 | komunikacja przy użyciu protokołu TCP |
| ARP | 3 | komunikat Dostępności |
| ARP | 4 | komunikat potwierdzający odebranie ARP - PID=3 i potwierdzenie gotowości rozmowy |

Podczas wysyłania wiadomości ARP-Request bądź ARP-Reply wykorzystywane są 42 bajty, pozostałe 18 wykorzystano w następujący sposób:



Każdy z ukrytych węzłów, który odebrał wiadomość ARP-Request zobligowany jest do analizy zawartości dopełnienia ramki Ethernetowej , a następnie wyliczenia skrótów *RH* wykorzystując ustaloną funkcję skrótu MD5 na bazie odebranych wartości: adresu MAC nadawcy (*SR\_MAC*), zawartości pola *RD* oraz ustalonych wartości:



Standardowo wpis w lokalnej tablicy ARP (ARP cache) hosta, jeśli nie został odświeżony, ulega usunięciu w przedziale od 1 do 20 minut w związku z tym każdy węzeł wysyła wiadomość rozgłoszeniową z informacją o swojej dostępności co 10 minut.

**Faza II – Inicjalizacja protokołu nośnika steganogramu**

Po uzyskaniu informacji o dostępności ukrytych węzłów zestawiamy połączenie z dostępnym użytkownikiem. Wykonujemy to w następującysposób:

* 1. Wysyłamy wiadomość ARP z odpowiednią wartością PID ( PID=1 – protokół TCP, PID=2 – protokół ICMP ) do dostępnego użytkownika, oznaczając wybór protokołu przez który będzie się komunikował z drugim rozmówcą.

* 1. Odbiorca po przechwyceniu ramki od nadawcy i akceptacji rozmowy wysyła wiadomość ARP z wartością PID oznaczając wybór protokołu przez który będzie się komunikował z drugim rozmówcą.

**Faza III – Ukryta wymiana danych**

W przypadku protokołu TCP musi nastąpić generacja pakietów TCP-ACK między użytkownikami. W związku z tym można przykładowo pobierać plik za pomocą protokołu FTP od użytkownika z którym użytkownik pobierający będzie się komunikował przez protokół TCP. Ukryta informacja umieszczona jest w ostatnich 6 Bajtach ramki zawierającą wiadomość typu TCP-ACK .

W przypadku protokołu ICMP nie ma konieczności generowania ruchu. Pakiety są wysyłane z częstością określoną w kolejnym punkcie. Wysyłane są wiadomości typu ICMP- Echo Request. Ukryta informacja umieszczona jest w ostatnich 18 Bajtach ramki zawierającą wiadomość typu ICMP – Echo Request .

Każdy użytkownik komunikatora posiada listę ukrytych węzłów : Nazwa odbiorcy oraz przypisany do niego adres IP. Informacje te przechowuje w pliku konfiguracyjnym. Posiada on informacje również o zestawianych połączeniach między innymi ukrytymi węzłami.

## Statystyka generowanego ruchu

Wybór protokołów służących do komunikacji, jak również częstość wysyłania informacji oparliśmy analizą ruchu sieci lokalnej zamieszczonej w artykule : „System steganograficzny oparty na niepoprawnym dopełnianiu ramek” opracowany przez Bartosza Jankowskiego, Wojciech Mazurczyka i Krzysztofa Szczypiorskiego.

Statystyka:

* Prawie 93% przechwyconych protokołów to protokoły bazujące na TCP
* Protokół ICMP - głównie wiadomości Echo Request i Echo Reply stanowi ok. 2.5% całkowitego ruchu
* Prawie 5% całkowitego ruchu stanowiły ramki z niepoprawnym dopełnieniem

Obliczenia:

Decyzje projektowe:

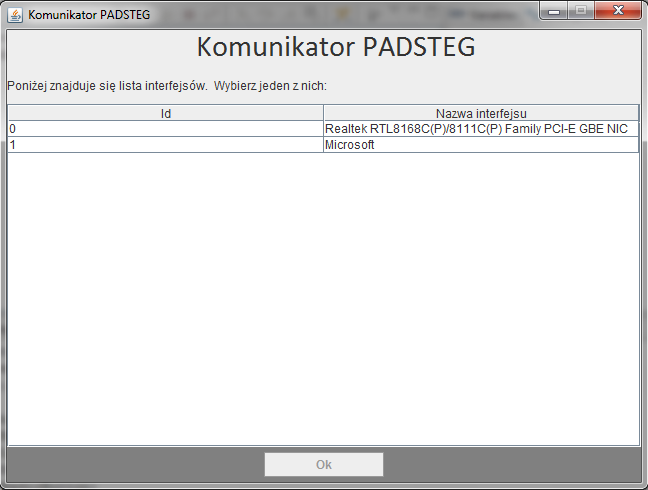
* Podczas rozmowy opartej o protokół TCP będziemy wysyłać pakiety z częstością: jedna wysłana ramka na 20 przechwyconych ramek typu Tcp-Ack.
* Podczas rozmowy opartej o protokół ICMP będziemy wysyłać pakiety z częstością: jedna wysłana ramka na 800 przechwyconych ( w tym przypadku nie ma ograniczenia co do przechwyconych pakietów).

# Opis aplikacji

Aplikacja została napisana w języku JAVA przy użyciu biblioteki SWING. W celu uzyskanie możliwości dostępu do ramek skorzystaliśmy z biblioteki jNetPcap, która jest „wrapperem” biblioteki LibPcap/WinPcap.

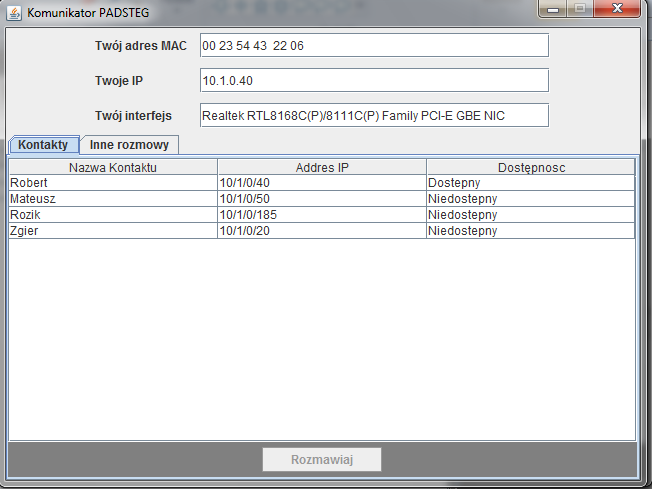
Po uruchomieniu aplikacji pojawia się następujący ekran:

**Ekran nr.1**



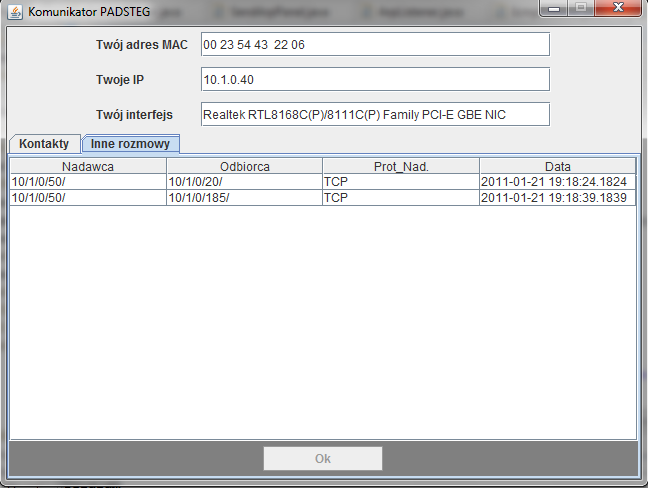
Ekran wyświetla wszystkie interfejsy kart sieciowych zainstalowane w danym komputerze. Po wyborze jednego z interfejsów należy zatwierdzić swoją decyzję przyciskiem OK. Przed przełączeniem się na następny ekran pojawi się komunikat, który umożliwia rezygnację z wyboru bądź jej zatwierdzenie. Akceptacja komunikatu spowoduje pojawienie się następującego ekranu.

**Ekran nr.2**



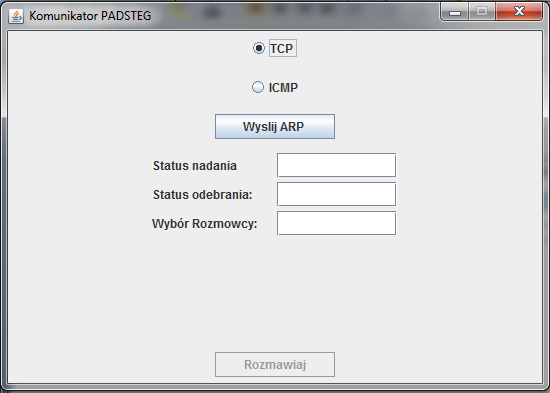
Ekran składa się z dwóch części. Pierwsza przedstawia parametry wybranej karty sieciowej, druga natomiast prezentuję tabele zawierająca listę użytkowników wczytaną z pliku konfiguracyjnego. Tabela zawiera trzy kolumny, z których godną szczególnej uwagi jest trzecia ponieważ wyświetla aktualny status użytkowników. Status „Dostępny” oznacza, że użytkownik posiada włączoną aplikację oraz jest gotowy na inicjację komunikacji. Zakładka „Inne rozmowy” przedstawia inne rozmowy prowadzone przez użytkowników z naszej listy.

**Ekran nr.3**



Po wybraniu użytkownika z listy a następnie naciśnięciu przycisku „Rozmawiaj”(Ekran nr.2) pojawia się następujący ekran.

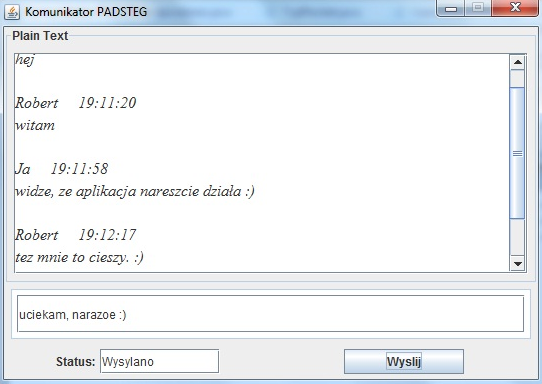
**Ekran nr.4**



Ekran przedstawia odpowiednio statusy nadania, odebrania oraz protokół, który został wybrany przez naszego rozmówcę. Umożliwia on również wybór protokołu po którym ma następować komunikacja, w naszym przypadku są to protokoły TCP oraz ICMP. Wybór naszego rozmówcy w żaden sposób nie wpływa na naszą decyzję, dla przykładu jedna strona może używać protokołu ICMP druga natomiast TCP.

Po wysłaniu a następnie odebraniu wiadomości ARP można rozpocząć rozmowę. Kliknięcie przycisku „Rozmawiaj” powoduje pojawienie się następującego ekranu.

**Ekran nr.5**

****

Jak widać ekran przedstawia główne okno rozmowy. Do wiadomości poszczególnych użytkowników dopisywane są odpowiednio: nazwa użytkownika oraz godzina wysłania wiadomości. Wysłanie wiadomości może być inicjowane przez wybranie przycisku „Wyslij” bądź przez naciśnięcie przycisku „Enter”.

# Testy

Podczas tworzenia oprogramowania komunikatora zostały przeprowadzone testy w sieci akademickiej, w akademiku D.S. Riviera.

1. Protokół TCP

W założeniach projektowych przyjęto zasadę, że pakiety z ukrytymi danymi wysyłane będą co 20 przechwyconych pakietów TCP- Ack. Testy przeprowadzone zostały przy wykorzystaniu protokołu FTP, który korzysta z protokołu TCP. Podczas obustronnego ściągania plików zostało wygenerowanych wiele pakietów TCP-Ack. Zapewniło to doskonałą płynność w wysyłaniu i odbieraniu wiadomości. Nie zauważono zbyt dużej różnicy między wysyłaniem ukrytych danych co 20 pakietów a wysyłaniem tych danych co 200 pakietów.

1. Protokół ICMP

W założeniach projektowych przyjęto zasadę, że pakiety z ukrytymi danymi wysyłane będą co 800 przechwyconych pakietów w sieci. Testy pokazały, iż korzystanie wyłącznie z tego protokołu nie zapewniło płynności w wysyłaniu wiadomości, pakiety były wysyłane relatywnie rzadko.

1. ICMP –TCP

Korzystanie z po jednej stronie z protokołu TCP a po drugiej z protokołu ICMP przyniosło zdecydowanie lepsze rezultaty. Testy przeprowadzono ponownie przy użyciu protokołu FTP. Dzięki generowaniu dodatkowego ruchu ( jeden z uczestników rozmowy ściągał plik od drugiej uczestnika) można było szybciej wysłać pakiety typu ICMP z ukrytymi danymi. Wysyłanie pakietów typu Tcp-Ack spowodowało niezmienną, bardzo dobrą dynamikę wysyłania wiadomości.

# Podsumowanie

Komunikator PadSteg okazał się bardzo dobrym narzędziem steganograficznym. Zaproponowany przez nas prototyp komunikatora opierający się o zasadę działania metody międzyprotokołowej steganografii sieciowej typu PadSteg przyniósł zadowalające rezultaty. Dobrym sposobem komunikacji okazuje się korzystanie z różnych protokołu : TCP- ICMP podczas pojedynczej rozmowy. Zmniejsza ono wykrywalność prowadzonej komunikacji, ponieważ zachowanie uczestników rozmowy imituje zachowanie zwykłego uczestnika sieci. Dzięki generowaniu dodatkowego ruchu związanego z wykorzystaniem protokołu FTP , który korzysta z potwierdzeń typu Tcp-Ack dynamika wysyłania pakietów typu ICMP jest zadowalająca. Korzystanie obustronne z protokołu TCP przynosi bardzo dobre rezultaty, lecz niestety zbyt częste korzystanie z tej metody może pomóc w wykryciu uczestników tajnej rozmowy, ponieważ zbyt częste obustronne przykładowo ściąganie plików w tym samym momencie może budzić podejrzenia.

Prototyp Komunikatora PadSteg jest narzędziem, który powinien w przyszłości być rozwijany. Wykorzystanie większej liczby protokołów i częste przełączanie umożliwi ograniczenie wykrywalności prowadzonej rozmowy do minimum.

# Bibliografia

1. D.E. Comer „Sieci komputerowe i intersieci”

2. RFC 792: TCP, RFC 793: ICMP

# 