

# BEST

-

## Projekt

Paweł Murdzek, 310850

Piotr Szewczyk, 311105

Politechnika Warszawska

13 czerwca 2025

## Spis treści

<b>1. Wstęp</b>	2
<b>2. Analiza Grup</b>	2
2.1. P1	2
2.1.1. Hipoteza 1: Steganografia w nazwach domen	2
2.1.2. Hipoteza 2: Ukrycie bajtów w polu Transaction ID	3
2.1.3. Hipoteza 3: Ukrycie znaków w ostatnim okcie adresu IP	3
2.1.4. Wnioski	3
2.2. P2	3
2.2.1. Analiza Potencjalnych Kanałów Steganograficznych	3
2.2.2. Identyfikacja Prawidłowego Nośnika Danych: Pole Identyfikacyjne IP ( <code>ip.id</code> )	4
2.2.3. Wnioski	4
2.3. P3	4
2.3.1. Hipoteza 1: Dekodowanie Pola <code>ip.id</code> jako Dwu-bajtowego Znak ASCII	4
2.3.2. Hipoteza 2: Dekodowanie Pola <code>tcp.payload</code>	5
2.3.3. Analiza LSB pól nagłówka TCP	5
2.4. P4	5
2.4.1. Etap 1: Analiza Wstępna i Falsyfikacja Hipotez	5
2.4.2. Etap 2: Identyfikacja Kanału Steganograficznego	6
2.4.3. Etap 3: Proces Dekodowania	6
2.4.4. Wyniki	6
2.4.5. Podsumowanie	6
2.5. P5	7
2.5.1. Hipoteza nr 1: Steganografia w polu danych aplikacji ( <code>payload</code> )	7
2.5.2. Hipoteza nr 2: Steganografia w kanałach ukrytych (nagłówki i kanał czasowy)	7
2.6. P6	7
2.6.1. Identyfikacja Kanału Ukrytego	7
2.6.2. Ekstrakcja Ukrytych Danych	8
2.6.3.	8
2.6.4. Testowanie Listy Prawdopodobnych Kluczy	8
2.6.5. Próba Ataku ze Znany Tekstem Jawnym	8
2.6.6. Wnioski	8
2.7. P7	9
2.7.1. Identyfikacja Kanału i Metody Kodowania	9
2.7.2. Przykłady Zakamuflowanych Danych	9
2.7.3. Schemat Metody Steganograficznej	9

## 1. Wstęp

Wstępna faza projektu polegała na próbie zastosowania zautomatyzowanego skryptu w języku Python w celu wykrycia potencjalnych kanałów steganograficznych. Skrypt realizował analizę w czterech kluczowych obszarach:

- **Analiza długości pakietów:** Badanie rozkładu długości pakietów pod kątem anomalii, w szczególności nierównej dystrybucji wartości parzystych i nieparzystych. Taka anomalia mogłaby wskazywać na ukrywanie danych w najmniej znaczących bitach (LSB) pól długości.
- **Analiza pól nagłówków TCP/IP:** Inspekcja pól takich jak znaczniki czasu TCP (TCP Timestamps) oraz pole identyfikacji IP (IP Identification Field) pod kątem niestandardowych sekwencji, wysokiej wariancji lub nienaturalnych wzorców LSB, które mogłyby świadczyć o iniekcji danych.
- **Analiza ładunku (payloadu) pakietów:** Ocena entropii ładunku, gdzie zarówno bardzo niska, jak i bardzo wysoka entropia mogą sugerować obecność ukrytych danych (np. zaszyfrowanych lub skompresowanych). Dodatkowo, analizowano częstotliwości występowania bajtów oraz poszukiwano predefiniowanych fraz (np. z tekstu "Antygony"), co mogłoby wskazywać na steganografię tekstową.
- **Analiza specyficznych pól protokołów (HTTP/DNS):** Szczegółowe badanie nagłówków HTTP (np. User-Agent, Referer, URI) oraz zapytań i odpowiedzi DNS (np. QNAME, typy rekordów TXT/NULL, entropia etykiet) w poszukiwaniu oznak tunelowania lub ukrywania informacji.

Zastosowanie powyższego zautomatyzowanego podejścia nie przyniosło oczekiwanych rezultatów i nie pozwoliło na identyfikację zastosowanych metod steganograficznych.

W kolejnym kroku podjęto próbę analizy statycznej i dynamicznej dostarczonego pliku wykonywalnego `.exe`, służącego do wykrywania steganografii, w celu poznania jego logiki działania. Analiza statyczna bootloadera C/C++ za pomocą narzędzia Ghidra okazała się nieefektywna, ponieważ komponent ten odpowiadał głównie za konfigurację środowiska uruchomieniowego, a nie za logikę detekcji. Próby dekompilacji wyodrębnionego kodu bajtowego Pythona w wersji 3.11 za pomocą narzędzi `uncompyle6` i `decompyle3` nie powiodły się z powodu braku wsparcia dla tej wersji, co uniemożliwiło odtworzenie kodu źródłowego. Równocześnie, ekstrakcja ciągów znaków z pliku wykonywalnego za pomocą `strings.exe` nie dała żadnych wyników, co sugeruje, że kluczowe dane tekstowe (w tym komunikat "Wklej ścieżkę pliku .pcap:") są zaciemnione lub dynamicznie dekompresowane w czasie wykonania. Wobec tych przeszkód, dalsze działania skoncentrowano na analizie ręcznej pakietów sieciowych

## 2. Analiza Grup

### 2.1. P1

41 0.042175	127.0.0.1	127.0.0.1	MDNS	97 Standard query response 0x3557 A outlook.microsoft.pl, "QH" question A 192.168.56.65
42 0.043036	127.0.0.1	127.0.0.1	MDNS	79 Standard query 0x5258 A login.microsoft.pl, "QH" question
43 0.044285	127.0.0.1	127.0.0.1	MDNS	95 Standard query response 0x5258 A login.microsoft.pl, "QH" question A 192.168.56.85
44 0.044978	127.0.0.1	127.0.0.1	MDNS	82 Standard query 0x2203 A onedrive.microsoft.pl, "QH" question
45 0.046260	127.0.0.1	127.0.0.1	MDNS	98 Standard query response 0x2203 A onedrive.microsoft.pl, "QH" question A 192.168.56.75
46 0.047042	127.0.0.1	127.0.0.1	MDNS	88 Standard query 0x6f2 A yammer.microsoft.pl, "QH" question
47 0.048563	127.0.0.1	127.0.0.1	MDNS	96 Standard query response 0x6f2 A yammer.microsoft.pl, "QH" question A 192.168.56.125
48 0.049203	127.0.0.1	127.0.0.1	MDNS	84 Standard query 0x31b9 A compliance.microsoft.pl, "QH" question
49 0.050761	127.0.0.1	127.0.0.1	MDNS	100 Standard query response 0x31b9 A compliance.microsoft.pl, "QH" question A 192.168.56.195
50 0.051476	127.0.0.1	127.0.0.1	MDNS	81 Standard query 0x2462 A outlook.microsoft.pl, "QH" question
51 0.052993	127.0.0.1	127.0.0.1	MDNS	97 Standard query response 0x2462 A outlook.microsoft.pl, "QH" question A 192.168.56.65
52 0.053698	127.0.0.1	127.0.0.1	MDNS	88 Standard query 0x57ee A portal.microsoft.pl, "QH" question
53 0.055012	127.0.0.1	127.0.0.1	MDNS	96 Standard query response 0x57ee A portal.microsoft.pl, "QH" question A 192.168.56.95
54 0.055041	127.0.0.1	127.0.0.1	MDNS	88 Standard query 0x71fb A portal.microsoft.pl, "QH" question
55 0.057201	127.0.0.1	127.0.0.1	MDNS	96 Standard query response 0x71fb A portal.microsoft.pl, "QH" question A 192.168.56.95
56 0.057857	127.0.0.1	127.0.0.1	MDNS	79 Standard query 0x4601 A azure.microsoft.pl, "QH" question
57 0.059152	127.0.0.1	127.0.0.1	MDNS	95 Standard query response 0x4601 A azure.microsoft.pl, "QH" question A 192.168.56.105
58 0.059065	127.0.0.1	127.0.0.1	MDNS	79 Standard query 0x7761 A skype.microsoft.pl, "QH" question
59 0.061307	127.0.0.1	127.0.0.1	MDNS	95 Standard query response 0x7761 A skype.microsoft.pl, "QH" question A 192.168.56.115
60 0.062047	127.0.0.1	127.0.0.1	MDNS	79 Standard query 0x77bd A azure.microsoft.pl, "QH" question
61 0.063456	127.0.0.1	127.0.0.1	MDNS	95 Standard query response 0x77bd A azure.microsoft.pl, "QH" question A 192.168.56.105
62 0.064056	127.0.0.1	127.0.0.1	MDNS	88 Standard query 0x71f3 A yammer.microsoft.pl, "QH" question
63 0.065263	127.0.0.1	127.0.0.1	MDNS	96 Standard query response 0x71f3 A yammer.microsoft.pl, "QH" question A 192.168.56.125
64 0.066810	127.0.0.1	127.0.0.1	MDNS	79 Standard query 0x1313 A azure.microsoft.pl, "QH" question

Rysunek 1: Ruch sieciowy grupy P1, charakteryzujący się obecnością zapytań mDNS.

#### 2.1.1. Hipoteza 1: Steganografia w nazwach domen

Analizę rozpoczęto od hipotezy dotyczącej ukrycia informacji w nazwach domen zapytań mDNS. Użycie litero-rozbioru "rn" zamiast "m" w nazwie "r.microsoft.pl" zinterpretowano jako potencjalną wskazówkę. Skupiono się na długościach etykiet w nazwach domen, zauważając częste występowanie etykiet o długości 10, co odpowiada znakowi nowej linii (<LF>). Mimo iż częstość występowania tej długości była znacząco wyższa niż w czystym ruchu sieciowym, analiza pozostałych długości (np. 5, 8) oraz próby dekodowania z użyciem operacji **XOR** nie doprowadziły do odcodowania spójnej wiadomości. Ta ścieżka analizy okazała się nieproduktywna.

### 2.1.2. Hipoteza 2: Ukrycie bajtów w polu Transaction ID

Kolejna hipoteza zakładała, że informacja jest zakodowana w 16-bitowym polu **Transaction ID** zapytań DNS. Po ekstrakcji wartości tego pola i porównaniu z oczekiwanym tekstem początkowym, zaobserwowano częściową zgodność.

Nr Ramki	Transaction ID (Hex)	Analizowany Bajt	Wartość Hex	Znak ASCII
3	674f	Drugi	4f	O
5	2065	Pierwszy	20	(spacja)
7	4472	-	75	u
9	06de	-	6b	k

Tabela 1: Próba dekodowania danych z pola Transaction ID.

Zgodność występowała jedynie dla pierwszych dwóch pakietów, co sugerowało, że przyjęta metoda jest błędna lub niekompletna. Hipoteza, że jeden z bajtów pola jest losowym szumem, nie znalazła potwierdzenia w dalszej analizie, co doprowadziło do jej odrzucenia.

### 2.1.3. Hipoteza 3: Ukrycie znaków w ostatnim okciecie adresu IP

Następnie zbadano możliwość ukrycia znaków ASCII w ostatnim okciecie adresu IP serwera DNS w pakietach odpowiedzi.

Nr Ramki	Adres IP w odpowiedzi DNS	Wartość ostatniego oktetu	Znak
14	192.168.56.65	65	A
16	192.168.56.115	115	s
18	192.168.56.185	185	ą

Tabela 2: Analiza steganograficzna adresów IP w odpowiedziach DNS.

Odkodowany ciąg znaków nie był zgodny z tekstem źródłowym. Dodatkowo, zaobserwowano, że wszystkie adresy IP kończą się cyfrą 5, jednak identyczna zależność występowała w pliku z czystym ruchem sieciowym. To potwierdziło, że jest to cecha konfiguracji sieci, a nie kanał steganograficzny. Hipoteza została sfalsyfikowana.

### 2.1.4. Wnioski

Pomimo wszechstronnej analizy obejmującej różne pola protokołu DNS, nie zidentyfikowano zastosowanej techniki steganograficznej w przypadku grupy P1.

## 2.2. P2

W analizowanym ruchu stwierdzono, że jako szum informacyjny (tzw. *red herring*) wykorzystano tekst "Lorem Ipsum", przesyłany w ładunku pakietów UDP.

### 2.2.1. Analiza Potencjalnych Kanałów Steganograficznych

**LSB długości pakietów** Postawiono hipotezę o kodowaniu informacji w najmniej znaczącym bicie (LSB) długości pakietów. Ekstrakcja i analiza sekwencji bitów LSB nie wykazała żadnej korelacji ani ukrytej wiadomości, co pozwoliło odrzucić tę hipotezę.

**Pole Danych UDP (udp.payload)** Analiza zawartości pola `udp.payload` ujawniła jedynie tekst "Lorem Ipsum", który zidentyfikowano jako element dezinformacyjny, mający na celu odwrócenie uwagi analityka.

**Suma Kontrolna UDP (udp.checksum)** Kolejnym badanym polem była suma kontrolna UDP. Stwierdzono, że pole `udp.checksum.status` dla wszystkich analizowanych pakietów posiadało status **2 (Unverified)**. Oznacza to, że wartość sumy kontrolnej nie była weryfikowana i jest niewiarygodna. Fakt ten ostatecznie wykluczył to pole jako wiarygodny nośnik informacji.

5664	321.783709	192.168.100.3	192.168.1.199	UDP	301 12345 → 12345 Len=255
5665	321.857943	192.168.100.3	192.168.1.199	UDP	336 12345 → 12345 Len=290
5666	321.956004	192.168.100.3	192.168.1.199	UDP	335 12345 → 12345 Len=289
5667	321.981239	192.168.100.3	192.168.1.199	UDP	217 12345 → 12345 Len=171
5668	322.025820	192.168.100.3	192.168.1.199	UDP	206 12345 → 12345 Len=160
5669	322.121401	192.168.100.3	192.168.1.199	UDP	101 12345 → 12345 Len=55
5670	322.185892	192.168.100.3	192.168.1.199	UDP	149 12345 → 12345 Len=103
5671	322.221481	192.168.100.3	192.168.1.199	UDP	373 12345 → 12345 Len=327
5672	322.278651	192.168.100.3	192.168.1.199	UDP	477 12345 → 12345 Len=431
5673	322.344601	192.168.100.3	192.168.1.199	UDP	173 12345 → 12345 Len=127
5674	322.445812	192.168.100.3	192.168.1.199	UDP	426 12345 → 12345 Len=380
5675	322.537560	192.168.100.3	192.168.1.199	UDP	141 12345 → 12345 Len=95
5676	322.601909	192.168.100.3	192.168.1.199	UDP	387 12345 → 12345 Len=341
5677	322.618695	192.168.100.3	192.168.1.199	UDP	148 12345 → 12345 Len=102
5678	322.694490	192.168.100.3	192.168.1.199	UDP	202 12345 → 12345 Len=156
5679	322.777323	192.168.100.3	192.168.1.199	UDP	263 12345 → 12345 Len=217
5680	322.869481	192.168.100.3	192.168.1.199	UDP	320 12345 → 12345 Len=274
5681	322.939852	192.168.100.3	192.168.1.199	UDP	432 12345 → 12345 Len=386
5682	323.037182	192.168.100.3	192.168.1.199	UDP	277 12345 → 12345 Len=231
5683	323.065374	192.168.100.3	192.168.1.199	UDP	145 12345 → 12345 Len=99
5684	323.142320	192.168.100.3	192.168.1.199	UDP	137 12345 → 12345 Len=91
5685	323.181883	192.168.100.3	192.168.1.199	UDP	460 12345 → 12345 Len=414
5686	323.270998	192.168.100.3	192.168.1.199	UDP	223 12345 → 12345 Len=177
5687	323.369635	192.168.100.3	192.168.1.199	UDP	383 12345 → 12345 Len=337
5688	323.456336	192.168.100.3	192.168.1.199	UDP	168 12345 → 12345 Len=122
5689	323.474905	192.168.100.3	192.168.1.199	UDP	316 12345 → 12345 Len=270
5690	323.571389	192.168.100.3	192.168.1.199	UDP	397 12345 → 12345 Len=351
5691	323.587132	192.168.100.3	192.168.1.199	UDP	236 12345 → 12345 Len=190
5692	323.643935	192.168.100.3	192.168.1.199	UDP	411 12345 → 12345 Len=365
5693	323.708020	192.168.100.3	192.168.1.199	UDP	183 12345 → 12345 Len=137

Rysunek 2: Ruch sieciowy grupy P2, zdominowany przez pakiety UDP na porcie 12345.

### 2.2.2. Identyfikacja Prawidłowego Nośnika Danych: Pole Identyfikacyjne IP (ip.id)

Po wykluczeniu powyższych hipotez, analiza skupiła się na 16-bitowym polu identyfikacyjnym nagłówka IP (ip.id). Zauważono, że jego wartości zmieniają się w każdym pakiecie. Postawiono hipotezę, że każda 16-bitowa wartość tego pola koduje dwa 8-bitowe znaki ASCII.

Proces dekodowania polegał na odczytaniu wartości pola ip.id, potraktowaniu jej jako dwóch bajtów i zdekodowaniu ich na znaki ASCII.

```
ip.id: 0x4162 -> "Ab"
ip.id: 0x7920 -> "y "
ip.id: 0x726f -> "ro"
ip.id: 0x7a70 -> "zp"
...i tak dalej.
```

### 2.2.3. Wnioski

Konkatenacja zdekodowanych fragmentów ze wszystkich pól ip.id w pliku pozwoliła na odtworzenie pełnej ukrytej wiadomości – fragmentu tekstu *Sofokles-Antygona.txt*. Zastosowana technika polegała na ukryciu danych w modyfikowalnym, lecz często pomijanym w analizie, polu nagłówka IP.

## 2.3. P3

W przypadku tej grupy przeprowadzono serię testów weryfikujących różne hipotezy dotyczące kanałów steganograficznych.

### 2.3.1. Hipoteza 1: Dekodowanie Pola ip.id jako Dwu-bajowego Znak ASCII

**Metoda** Technika ta zakładała, że 16-bitowe pole identyfikacyjne IP (ip.id) jest wykorzystywane do przenoszenia dwóch 8-bitowych znaków ASCII. Wartość heksadecymalna pola, np. 0x4142, jest dzielona na dwa bajty (41 i 42), a następnie każdy z nich jest konwertowany na odpowiadający mu znak ASCII ('A' i 'B').

$$C_1, C_2 = \text{decode}(\text{hex\_to\_bytes}(\text{ip.id}))$$

**Wynik** Analiza pliku tą metodą nie ujawniła ukrytego tekstu. Zdekodowane dane składały się z nieczytelnych znaków oraz wartości zerowych, nie tworząc żadnej logicznej wiadomości. Hipoteza została sfalsyfikowana.

1	0.00000000	172.30.112.127	172.30.112.126	TCP	54 46302 → 80 [SYN] Seq=0 Win=0 Len=0
2	0.000048227	172.30.112.126	172.30.112.127	TCP	58 80 → 46302 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
3	0.000052374	172.30.112.127	172.30.112.126	TCP	54 46302 → 80 [RST] Seq=0 Win=0 Len=0
4	0.037318102	172.30.112.126	172.30.112.127	TCP	54 [TCP Previous segment not captured] 80 → 46302 [ACK] Seq=1116201890 Ack=1 Win=0 Len=0
5	0.037994919	172.30.112.127	172.30.112.126	TCP	54 46302 → 80 [RST] Seq=1 Win=0 Len=0
6	0.058185186	172.30.112.127	172.30.112.126	TCP	74 29225 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=4130481170 TSecr=0 W=128
7	0.058278832	172.30.112.126	172.30.112.127	TCP	74 80 → 29225 [SYN, ACK] Seq=0 Ack=1 Win=65152 Len=0 MSS=1460 SACK_PERM TSval=1798473274 TSecr=4130481170 W=128
8	0.058481875	172.30.112.126	172.30.112.127	TCP	54 80 → 46302 [RST] Seq=2497547056 Win=0 Len=0
9	0.058688816	172.30.112.127	172.30.112.126	TCP	66 29225 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=4130481171 TSecr=1798473271
10	0.058688877	172.30.112.127	172.30.112.126	HTTP	157 GET /telekom/api/support/tickets/open HTTP/1.1
11	0.058681216	172.30.112.126	172.30.112.127	TCP	66 80 → 29225 [ACK] Seq=1 Ack=92 Win=65152 Len=0 TSval=1798473272 TSecr=4130481171
12	0.061313391	172.30.112.126	172.30.112.127	TCP	233 80 → 29225 [PSH, ACK] Seq=1 Ack=92 Win=65152 Len=167 TSval=1798473274 TSecr=4130481171 [TCP PDU reassembled in 14]
13	0.061547547	172.30.112.127	172.30.112.126	TCP	80 29225 → 80 [ACK] Seq=92 Ack=168 Win=64128 Len=0 TSval=4130481174 TSecr=1798473274
14	0.061546958	172.30.112.126	172.30.112.127	HTTP/1.1	180 HTTP/1.1 200 OK, 350M (application/json)
15	0.061647757	172.30.112.127	172.30.112.126	TCP	66 29225 → 80 [ACK] Seq=92 Ack=282 Win=64128 Len=0 TSval=4130481174 TSecr=1798473274
16	0.071493843	172.30.112.126	172.30.112.127	TCP	66 80 → 29225 [FIN, ACK] Seq=202 Ack=92 Win=65152 Len=0 TSval=1798473284 TSecr=4130481174
17	0.072519130	172.30.112.127	172.30.112.126	TCP	66 29225 → 80 [FIN, ACK] Seq=92 Ack=203 Win=64128 Len=0 TSval=4130481195 TSecr=1798473284
18	0.072558489	172.30.112.126	172.30.112.127	TCP	66 80 → 29225 [ACK] Seq=283 Ack=93 Win=65152 Len=0 TSval=1798473285 TSecr=4130481195
19	0.082661749	172.30.112.127	172.30.112.126	TCP	54 43959 → 80 [SYN] Seq=0 Win=0 Len=0
20	0.082721292	172.30.112.126	172.30.112.127	TCP	58 80 → 43959 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
21	0.083051544	172.30.112.127	172.30.112.126	TCP	54 43959 → 80 [RST] Seq=1 Win=0 Len=0
22	0.090578385	172.30.112.126	172.30.112.127	TCP	54 [TCP Previous segment not captured] 80 → 43959 [ACK] Seq=1155455553 Ack=1 Win=0 Len=0
23	0.091067442	172.30.112.127	172.30.112.126	TCP	54 43959 → 80 [RST] Seq=1 Win=0 Len=0
24	0.111944673	172.30.112.126	172.30.112.127	TCP	54 80 → 43959 [RST] Seq=778053979 Win=0 Len=0
25	0.130846484	172.30.112.127	172.30.112.126	TCP	74 4189 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=4130481251 TSecr=0 W=128
26	0.139212968	172.30.112.126	172.30.112.127	TCP	74 80 → 4189 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=1798473352 TSecr=4130481251 W=128
27	0.139702748	172.30.112.127	172.30.112.126	TCP	66 4189 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=4130481252 TSecr=1798473352
28	0.139782822	172.30.112.127	172.30.112.126	HTTP	143 GET /telekom/api/status HTTP/1.1
29	0.139854642	172.30.112.126	172.30.112.127	TCP	66 80 → 4189 [ACK] Seq=1 Ack=78 Win=65152 Len=0 TSval=1798473353 TSecr=4130481252
30	0.142083833	172.30.112.126	172.30.112.127	TCP	232 80 → 4189 [PSH, ACK] Seq=1 Ack=78 Win=65152 Len=166 TSval=1798473355 TSecr=4130481252 [TCP PDU reassembled in 32]
31	0.142081345	172.30.112.127	172.30.112.126	TCP	66 4189 → 80 [ACK] Seq=78 Ack=167 Win=64128 Len=0 TSval=4130481255 TSecr=1798473355
32	0.142723488	172.30.112.126	172.30.112.127	HTTP/1.1	181 HTTP/1.1 200 OK, 350M (application/json)
33	0.143538811	172.30.112.127	172.30.112.126	TCP	66 4189 → 80 [ACK] Seq=78 Ack=282 Win=64128 Len=0 TSval=4130481255 TSecr=1798473355
34	0.152036618	172.30.112.126	172.30.112.127	TCP	66 80 → 4189 [FIN, ACK] Seq=202 Ack=78 Win=65152 Len=0 TSval=1798473365 TSecr=4130481255
35	0.153599772	172.30.112.127	172.30.112.126	TCP	66 4189 → 80 [FIN, ACK] Seq=78 Ack=203 Win=64128 Len=0 TSval=4130481265 TSecr=1798473365
36	0.153481458	172.30.112.126	172.30.112.127	TCP	66 80 → 4189 [ACK] Seq=203 Ack=79 Win=65152 Len=0 TSval=1798473366 TSecr=4130481265
37	0.155484411	172.30.112.127	172.30.112.126	TCP	54 42376 → 80 [SYN] Seq=0 Win=0 Len=0
38	0.155527099	172.30.112.126	172.30.112.127	TCP	58 80 → 42376 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
39	0.166303395	172.30.112.127	172.30.112.126	TCP	54 42376 → 80 [RST] Seq=1 Win=0 Len=0

Rysunek 3: Ruch sieciowy grupy P3, zawierający komunikację HTTP.

### 2.3.2. Hipoteza 2: Dekodowanie Pola tcp.payload

**Metoda** Ta metoda polegała na potraktowaniu zawartości pola `tcp.payload` jako surowych danych, które połączeniu i zdekodowaniu (UTF-8) mogłyby ujawnić ukrytą wiadomość.

**Wynik** Zdekodowana zawartość okazała się być standardowym ruchem sieciowym HTTP, zawierającym czytelne zapytanie (GET /telekom/api/status HTTP/1.1) i odpowiedzi serwera. Nie odnaleziono w niej tekstu z pliku Sofokles-Antygona.txt. Hipoteza została sfalsyfikowana.

### 2.3.3. Analiza LSB pól nagłówka TCP

Przeprowadzono analizę najmniej znaczącego bitu (LSB) dla kilku pól nagłówka TCP. Metoda polegała na ekstrakcji ostatniego bitu z wartości danego pola w każdym pakiecie, łączeniu bitów w 8-bitowe bajty i konwersji na znaki ASCII.

— **LSB Czasu TCP** (`tcp.options.timestamp.tsval`):

$$b_i = tsval_i \pmod{2}$$

Wynikiem tej analizy był ciąg losowych, nieczytelnych znaków.

— **LSB Numeru Sekwencyjnego TCP** (`tcp.seq_raw`):

$$b_i = seq\_raw_i \pmod{2}$$

Zdekodowana wiadomość była bezsensownym ciągiem znaków.

— **LSB Numeru Potwierdzenia TCP** (`tcp.ack_raw`):

$$b_i = ack\_raw_i \pmod{2}$$

Wynik był niekoherentnym zbiorem znaków.

— **LSB Rozmiaru Okna TCP** (`tcp.window_size_value`):

$$b_i = window\_size_i \pmod{2}$$

Otrzymany ciąg danych był wysoce powtarzalnym, bezsensownym wzorcem. Wszystkie hipotezy dotyczące steganografii LSB w polach nagłówka TCP zostały sfalsyfikowane.

## 2.4. P4

### 2.4.1. Etap 1: Analiza Wstępna i Falsyfikacja Hipotez

Początkowe działania analityczne skupiono na dwóch hipotezach, które okazały się elementami dezinformacyjnymi.



39	1.721788	127.0.0.1	127.0.0.1	TCP	44	56884 → 15201	[ACK] Seq=38 Ack=158 Win=327168 Len=0
40	1.721899	127.0.0.1	127.0.0.1	TCP	56	56886 → 56885	[SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
41	1.721937	127.0.0.1	127.0.0.1	TCP	56	56885 → 56886	[SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
42	1.721934	127.0.0.1	127.0.0.1	TCP	44	56886 → 56885	[ACK] Seq=1 Ack=1 Win=327424 Len=0
43	1.721949	127.0.0.1	127.0.0.1	TCP	165	56884 → 15201	[PSH, ACK] Seq=38 Ack=158 Win=327168 Len=121
44	1.721953	127.0.0.1	127.0.0.1	TCP	44	15201 → 56884	[ACK] Seq=158 Ack=159 Win=2161152 Len=0
45	1.726987	127.0.0.1	127.0.0.1	TCP	98	15201 → 56884	[PSH, ACK] Seq=158 Ack=159 Win=2161152 Len=54
46	1.726915	127.0.0.1	127.0.0.1	TCP	44	56884 → 15201	[ACK] Seq=159 Ack=212 Win=327168 Len=0
47	1.726957	127.0.0.1	127.0.0.1	TCP	362	56886 → 56885	[PSH, ACK] Seq=1 Ack=1 Win=327424 Len=318
48	1.726966	127.0.0.1	127.0.0.1	TCP	44	56885 → 56886	[ACK] Seq=1 Ack=319 Win=2160896 Len=0
49	1.726985	127.0.0.1	127.0.0.1	TCP	44	56886 → 56885	[FIN, ACK] Seq=319 Ack=1 Win=327424 Len=0
50	1.726990	127.0.0.1	127.0.0.1	TCP	44	56885 → 56886	[ACK] Seq=1 Ack=320 Win=2160896 Len=0
51	1.727034	127.0.0.1	127.0.0.1	TCP	44	56885 → 56886	[FIN, ACK] Seq=1 Ack=320 Win=2160896 Len=0
52	1.727042	127.0.0.1	127.0.0.1	TCP	44	56886 → 56885	[ACK] Seq=320 Ack=2 Win=327424 Len=0
53	1.727168	127.0.0.1	127.0.0.1	TCP	68	15201 → 56884	[PSH, ACK] Seq=212 Ack=159 Win=2161152 Len=24
54	1.727173	127.0.0.1	127.0.0.1	TCP	44	56884 → 15201	[ACK] Seq=159 Ack=236 Win=327168 Len=0
55	1.727545	127.0.0.1	127.0.0.1	TCP	52	56884 → 15201	[PSH, ACK] Seq=159 Ack=236 Win=327168 Len=8
56	1.727552	127.0.0.1	127.0.0.1	TCP	44	15201 → 56884	[ACK] Seq=236 Ack=167 Win=2161152 Len=0
57	1.727583	127.0.0.1	127.0.0.1	TCP	70	15201 → 56884	[PSH, ACK] Seq=236 Ack=167 Win=2161152 Len=26
58	1.727598	127.0.0.1	127.0.0.1	TCP	44	56884 → 15201	[ACK] Seq=167 Ack=262 Win=327168 Len=0
59	1.727603	127.0.0.1	127.0.0.1	TCP	98	56884 → 15201	[PSH, ACK] Seq=167 Ack=262 Win=327168 Len=6
60	1.727687	127.0.0.1	127.0.0.1	TCP	44	15201 → 56884	[ACK] Seq=262 Ack=173 Win=2161152 Len=0
61	1.727694	127.0.0.1	127.0.0.1	TCP	91	15201 → 56884	[PSH, ACK] Seq=262 Ack=173 Win=2161152 Len=47
62	1.727703	127.0.0.1	127.0.0.1	TCP	44	56884 → 15201	[ACK] Seq=173 Ack=389 Win=320912 Len=0
63	1.727708	127.0.0.1	127.0.0.1	TCP	56	56886 → 56887	[SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
64	1.727781	127.0.0.1	127.0.0.1	TCP	56	56887 → 56888	[SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
65	1.727792	127.0.0.1	127.0.0.1	TCP	44	56888 → 56887	[ACK] Seq=1 Ack=1 Win=2161152 Len=0
66	1.727806	127.0.0.1	127.0.0.1	TCP	204	56884 → 15201	[PSH, ACK] Seq=173 Ack=389 Win=320912 Len=160
67	1.727809	127.0.0.1	127.0.0.1	TCP	44	15201 → 56884	[ACK] Seq=389 Ack=333 Win=2160896 Len=0
68	1.728788	127.0.0.1	127.0.0.1	TCP	98	15201 → 56884	[PSH, ACK] Seq=389 Ack=333 Win=2160896 Len=54
69	1.728715	127.0.0.1	127.0.0.1	TCP	44	56884 → 15201	[ACK] Seq=333 Ack=383 Win=320912 Len=0
70	1.728751	127.0.0.1	127.0.0.1	TCP	362	56888 → 56887	[PSH, ACK] Seq=1 Ack=1 Win=2161152 Len=318
71	1.728768	127.0.0.1	127.0.0.1	TCP	44	56887 → 56888	[ACK] Seq=1 Ack=319 Win=2160896 Len=0
72	1.728771	127.0.0.1	127.0.0.1	TCP	44	56888 → 56887	[FIN, ACK] Seq=319 Ack=1 Win=2161152 Len=0

Rysunek 4: Ruch sieciowy grupy P4, zawierający sesje FTP i zapytania HTTP POST.

- **Hipoteza o danych w ładunku FTP:** Zidentyfikowano wielokrotne przesyłanie plików, których zawartość stanowiła znany tekst (tzw. cypypasta). Ze względu na powtarzalność i brak związku z tekstem źródłowym, stwierdzono, że ładunek ten pełni rolę szumu informacyjnego.
- **Hipoteza o kanale czasowym:** Zbadano statystyczne właściwości ruchu sieciowego, w tym rozkład bajtów oraz odstępy czasowe między pakietami (`frame.time_delta`). Analiza opóźnień wykazała występowanie dwóch wyraźnych grup, jednak hipoteza o kodowaniu informacji w czasie (ang. *\*timing channel\**) nie doprowadziła do odkodowania literalnego fragmentu tekstu.

#### 2.4.2. Etap 2: Identyfikacja Kanału Steganograficznego

Przełom w analizie nastąpił po przyjęciu hipotezy, że ukryta wiadomość jest zakodowana w innym, widocznym elemencie ruchu sieciowego. Uwagę skierowano na **nazwy plików** przesyłanych w ramach sesji FTP. W pakietach kontrolnych protokołu FTP (strumień TCP nr 5 i kolejne) zidentyfikowano serię komend **STOR**. Każda z nich zawierała unikalną, długą i pozornie losową nazwę pliku, jak w przykładzie z pakietu 43:

```
STOR 9640bb1.../JZWVM2KOKRCSWSJSJ...JFZT2.txt
```

Wysoka zmienność tych nazw sugerowała, że to one, a nie zawartość plików, są nośnikiem informacji.

#### 2.4.3. Etap 3: Proces Dekodowania

Proces odtworzenia wiadomości zrealizowano w następujących krokach:

1. **Ekstrakcja fragmentów:** Z każdej komendy **STOR** wyodrębniono ciąg znaków stanowiący nazwę pliku (bez rozszerzenia i ścieżki).
2. **Konkatenacja:** Wszystkie wyodrębnione ciągi połączono w jedną, długą sekwencję.
3. **Identyfikacja kodowania:** Połączony ciąg znaków wykazywał cechy kodowania **Base32** (alfabet A-Z, cyfry 2-7).
4. **Dekodowanie Base32:** Cały ciąg został zdekodowany z formatu Base32 do postaci binarnej.
5. **Konwersja na tekst:** Otrzymane dane binarne zinterpretowano jako tekst w kodowaniu **UTF-8**, co umożliwiło poprawne odczytanie polskich znaków diakrytycznych.

#### 2.4.4. Wyniki

Nie udało się poprawnie odszyfrować tego tekstu szyfrowaniem Base64 oraz Base32.

#### 2.4.5. Podsumowanie

Postawiona hipoteza o metodzie steganograficznej polegającej na zakodowaniu wiadomości za pomocą algorytmu Base32, a następnie jej fragmentacji i ukryciu w nazwach plików przesyłanych podczas sesji FTP się nie sprawdziła.

4	0.22809	102.168.0.139	102.168.0.137	FTP-DL	111 [TCP Previous segment not captured] FTP Data: 57 bytes
5	0.53018	102.168.0.137	102.168.0.139	FTP-DL	224 [TCP Previous segment not captured] FTP Data: 170 bytes
6	0.53145	102.168.0.139	102.168.0.137	FTP-DL	111 [TCP Acked unseq segment] [TCP Previous segment not captured] [TCP Spurious Retransmission] FTP Data: 57 bytes
7	0.73182	102.168.0.137	102.168.0.139	TCP	224 [TCP Spurious Retransmission] 20 = 80 [PSH, ACK] Seq=108378303 Ack=140432 Len=170
8	0.74770	102.168.0.139	102.168.0.137	FTP-DL	111 [TCP Previous segment not captured] [TCP Spurious Retransmission] FTP Data: 57 bytes
9	0.84212	102.168.0.137	102.168.0.139	TCP	224 [TCP Spurious Retransmission] 20 = 80 [PSH, ACK] Seq=551455957 Ack=140432 Len=170
10	0.93944	102.168.0.139	102.168.0.137	TCP	224 [TCP Spurious Retransmission] 20 = 80 [PSH, ACK] Seq=551455958 Ack=140432 Len=170
11	1.26437	102.168.0.137	102.168.0.139	TCP	224 [TCP Retransmission] 20 = 80 [PSH, ACK] Seq=425444008 Ack=140432 Len=170
12	0.47083	102.168.0.139	102.168.0.137	TCP	111 [TCP Acked unseq segment] [TCP Spurious Retransmission] 80 = 20 [PSH, ACK] Seq=551455958 Ack=140432 Len=170
13	1.47553	102.168.0.137	102.168.0.139	TCP	224 [TCP Retransmission] 20 = 80 [PSH, ACK] Seq=40840218 Ack=140432 Len=170
14	1.49277	102.168.0.139	102.168.0.137	FTP-DL	111 [TCP Previous segment not captured] FTP Data: 57 bytes
15	1.68012	102.168.0.137	102.168.0.139	TCP	224 [TCP Retransmission] 20 = 80 [PSH, ACK] Seq=551455959 Ack=140432 Len=170
16	1.71839	102.168.0.139	102.168.0.137	TCP	111 [TCP Retransmission] 80 = 20 [PSH, ACK] Seq=551455958 Ack=140432 Len=170
17	1.80077	102.168.0.137	102.168.0.139	FTP-DL	111 [TCP Previous segment not captured] [TCP Spurious Retransmission] FTP Data: 57 bytes
18	1.12747	102.168.0.137	102.168.0.139	FTP-DL	224 [TCP Previous segment not captured] FTP Data: 170 bytes
19	1.15219	102.168.0.139	102.168.0.137	TCP	111 [TCP Acked unseq segment] [TCP Retransmission] 80 = 20 [PSH, ACK] Seq=551455958 Ack=140432 Len=170
20	1.51614	102.168.0.137	102.168.0.139	TCP	224 [TCP Previous segment not captured] [TCP Spurious Retransmission] FTP Data: 170 bytes
21	1.62022	102.168.0.139	102.168.0.137	TCP	111 [TCP Acked unseq segment] [TCP Spurious Retransmission] 80 = 20 [PSH, ACK] Seq=551455958 Ack=140432 Len=170
22	1.66874	102.168.0.137	102.168.0.139	TCP	224 [TCP Spurious Retransmission] 20 = 80 [PSH, ACK] Seq=551455958 Ack=140432 Len=170
23	1.67555	102.168.0.139	102.168.0.137	TCP	111 [TCP Retransmission] 80 = 20 [PSH, ACK] Seq=551455958 Ack=140432 Len=170
24	1.80083	102.168.0.137	102.168.0.139	TCP	224 [TCP Retransmission] 20 = 80 [PSH, ACK] Seq=551455958 Ack=140432 Len=170
25	1.90359	102.168.0.139	102.168.0.137	TCP	111 [TCP Retransmission] 80 = 20 [PSH, ACK] Seq=551455958 Ack=140432 Len=170
26	1.11292	102.168.0.137	102.168.0.139	TCP	224 [TCP Spurious Retransmission] 20 = 80 [PSH, ACK] Seq=551455958 Ack=140432 Len=170
27	1.13784	102.168.0.139	102.168.0.137	FTP-DL	111 [TCP Previous segment not captured] [TCP Spurious Retransmission] FTP Data: 57 bytes
28	1.35464	102.168.0.137	102.168.0.139	FTP-DL	111 [TCP Acked unseq segment] [TCP Previous segment not captured] FTP Data: 57 bytes
29	1.44908	102.168.0.137	102.168.0.139	TCP	224 [TCP Retransmission] 20 = 80 [PSH, ACK] Seq=551455958 Ack=140432 Len=170
30	1.65180	102.168.0.139	102.168.0.137	TCP	111 [TCP Acked unseq segment] [TCP Spurious Retransmission] 80 = 20 [PSH, ACK] Seq=551455958 Ack=140432 Len=170
31	1.82886	102.168.0.137	102.168.0.139	FTP-DL	224 [TCP Previous segment not captured] [TCP Spurious Retransmission] FTP Data: 170 bytes
32	1.87861	102.168.0.139	102.168.0.137	TCP	111 [TCP Retransmission] 80 = 20 [PSH, ACK] Seq=551455958 Ack=140432 Len=170
33	1.84887	102.168.0.139	102.168.0.137	TCP	111 [TCP Acked unseq segment] [TCP Spurious Retransmission] 80 = 20 [PSH, ACK] Seq=551455958 Ack=140432 Len=170
34	1.92810	102.168.0.137	102.168.0.139	FTP-DL	224 [TCP Previous segment not captured] [TCP Spurious Retransmission] FTP Data: 170 bytes
35	1.93389	102.168.0.139	102.168.0.137	TCP	111 [TCP Retransmission] 80 = 20 [PSH, ACK] Seq=551455958 Ack=140432 Len=170
36	1.54801	102.168.0.137	102.168.0.139	TCP	224 [TCP Spurious Retransmission] 20 = 80 [PSH, ACK] Seq=551455958 Ack=140432 Len=170
37	1.56989	102.168.0.139	102.168.0.137	TCP	111 [TCP Acked unseq segment] [TCP Previous segment not captured] FTP Data: 57 bytes
38	1.78478	102.168.0.139	102.168.0.137	TCP	111 [TCP Acked unseq segment] [TCP Retransmission] 80 = 20 [PSH, ACK] Seq=551455958 Ack=140432 Len=170
39	1.86636	102.168.0.137	102.168.0.139	FTP-DL	224 [TCP Previous segment not captured] [TCP Spurious Retransmission] FTP Data: 170 bytes
40	1.87051	102.168.0.139	102.168.0.137	FTP-DL	111 [TCP Previous segment not captured] [TCP Spurious Retransmission] FTP Data: 170 bytes

Rysunek 5: Ruch grupy P5: sekwencja zapytań HTTP PUT aktualizująca zasób TICKET-123.

## 2.5. P5

### 2.5.1. Hipoteza nr 1: Steganografia w polu danych aplikacji (payload)

- **Założenie:** Ukryta wiadomość została osadzona bezpośrednio w polu `tcp.payload`.
- **Metodologia:** Zautomatyzowana ekstrakcja i analiza zawartości wszystkich pól `tcp.payload` pod kątem ciągów znaków ASCII/UTF-8 oraz słów kluczowych.
- **Wynik: Hipoteza sfalsyfikowana.** W polach `payload` zidentyfikowano wyłącznie standardową komunikację protokołu HTTP. Nie znaleziono żadnych fragmentów poszukiwanego tekstu.

### 2.5.2. Hipoteza nr 2: Steganografia w kanałach ukrytych (nagłówki i kanał czasowy)

- **Założenie:** Informacja została ukryta w polach nagłówków (np. `ip.id`, `tcp.seq`) lub poprzez kodowanie w odstępach czasu między pakietami.
- **Metodologia:** Analiza statystyczna wartości pól nagłówków oraz analiza rozkładu opóźnień między pakietami (`frame.time_delta`).
- **Wynik: Hipoteza sfalsyfikowana.** Analiza statystyczna nie wykazała żadnych anomalii. Wartości w nagłówkach oraz charakterystyka czasowa transmisji były zgodne ze standardowym działaniem stosu TCP/IP.

W przypadku tej grupy przeprowadzone analizy nie doprowadziły do odkrycia metody steganograficznej.

## 2.6. P6

319	23.08068	10.0.2.15	151.101.130.132	TCP	54 38864 → 80 [ACK] Seq=8924 Ack=503440 Win=65535 Len=0
320	23.080917	10.0.2.15	151.101.130.132	TCP	54 38864 → 80 [ACK] Seq=8924 Ack=512880 Win=65535 Len=0
321	23.080933	151.101.130.132	10.0.2.15	TCP	2934 80 → 38864 [PSH, ACK] Seq=536568 Ack=8924 Win=65535 Len=2880 [TCP PDU reassembled in 369]
322	23.080933	151.101.130.132	10.0.2.15	TCP	1494 80 → 38864 [PSH, ACK] Seq=536568 Ack=8924 Win=65535 Len=1440 [TCP PDU reassembled in 369]
323	23.080941	151.101.130.132	10.0.2.15	TCP	2934 80 → 38864 [PSH, ACK] Seq=516480 Ack=8924 Win=65535 Len=2880 [TCP PDU reassembled in 369]
324	23.080942	151.101.130.132	10.0.2.15	TCP	2934 80 → 38864 [PSH, ACK] Seq=515288 Ack=8924 Win=65535 Len=2880 [TCP PDU reassembled in 369]
325	23.080943	151.101.130.132	10.0.2.15	TCP	2934 80 → 38864 [PSH, ACK] Seq=522168 Ack=8924 Win=65535 Len=2880 [TCP PDU reassembled in 369]
326	23.081020	10.0.2.15	151.101.130.132	TCP	54 38864 → 80 [ACK] Seq=8924 Ack=525040 Win=65535 Len=0
327	23.082971	151.101.130.132	10.0.2.15	TCP	1494 80 → 38864 [PSH, ACK] Seq=525840 Ack=8924 Win=65535 Len=1440 [TCP PDU reassembled in 369]
328	23.082972	151.101.130.132	10.0.2.15	TCP	1514 80 → 38864 [PSH, ACK] Seq=526480 Ack=8924 Win=65535 Len=1440 [TCP PDU reassembled in 369]
329	23.082972	151.101.130.132	10.0.2.15	TCP	1287 80 → 38864 [PSH, ACK] Seq=527948 Ack=8924 Win=65535 Len=1233 [TCP PDU reassembled in 369]
330	23.082972	151.101.130.132	10.0.2.15	TCP	2122 80 → 38864 [PSH, ACK] Seq=529181 Ack=8924 Win=65535 Len=1967 [TCP PDU reassembled in 369]
331	23.082973	151.101.130.132	10.0.2.15	TCP	2934 80 → 38864 [PSH, ACK] Seq=532248 Ack=8924 Win=65535 Len=2880 [TCP PDU reassembled in 369]
332	23.083023	10.0.2.15	151.101.130.132	TCP	54 38864 → 80 [ACK] Seq=8924 Ack=527948 Win=65535 Len=0
333	23.083237	10.0.2.15	151.101.130.132	TCP	54 38864 → 80 [ACK] Seq=8924 Ack=532248 Win=65535 Len=0
334	23.083252	151.101.130.132	10.0.2.15	TCP	1494 80 → 38864 [PSH, ACK] Seq=535312 Ack=8924 Win=65535 Len=1440 [TCP PDU reassembled in 369]
335	23.083253	151.101.130.132	10.0.2.15	TCP	5814 80 → 38864 [PSH, ACK] Seq=536568 Ack=8924 Win=65535 Len=5760 [TCP PDU reassembled in 369]
336	23.083253	151.101.130.132	10.0.2.15	TCP	2934 80 → 38864 [PSH, ACK] Seq=543328 Ack=8924 Win=65535 Len=2880 [TCP PDU reassembled in 369]
337	23.083254	151.101.130.132	10.0.2.15	TCP	2934 80 → 38864 [PSH, ACK] Seq=545280 Ack=8924 Win=65535 Len=2880 [TCP PDU reassembled in 369]
338	23.083291	10.0.2.15	151.101.130.132	TCP	54 38864 → 80 [ACK] Seq=8924 Ack=548880 Win=65535 Len=0
339	23.083402	151.101.130.132	10.0.2.15	TCP	2934 80 → 38864 [PSH, ACK] Seq=548880 Ack=8924 Win=65535 Len=2880 [TCP PDU reassembled in 369]
340	23.083483	151.101.130.132	10.0.2.15	TCP	2934 80 → 38864 [PSH, ACK] Seq=550968 Ack=8924 Win=65535 Len=2880 [TCP PDU reassembled in 369]
341	23.083527	10.0.2.15	151.101.130.132	TCP	54 38864 → 80 [ACK] Seq=8924 Ack=550968 Win=65535 Len=0
342	23.083990	10.0.2.15	151.101.130.132	TCP	54 38864 → 80 [ACK] Seq=8924 Ack=553840 Win=65535 Len=0
343	23.084581	151.101.130.132	10.0.2.15	TCP	1494 80 → 38864 [PSH, ACK] Seq=553840 Ack=8924 Win=65535 Len=1440 [TCP PDU reassembled in 369]
344	23.084582	151.101.130.132	10.0.2.15	TCP	2934 80 → 38864 [PSH, ACK] Seq=555288 Ack=8924 Win=65535 Len=2880 [TCP PDU reassembled in 369]
345	23.084582	151.101.130.132	10.0.2.15	TCP	2934 80 → 38864 [PSH, ACK] Seq=556168 Ack=8924 Win=65535 Len=2880 [TCP PDU reassembled in 369]
346	23.084582	151.101.130.132	10.0.2.15	TCP	2934 80 → 38864 [PSH, ACK] Seq=561840 Ack=8924 Win=65535 Len=2880 [TCP PDU reassembled in 369]
347	23.084583	151.101.130.132	10.0.2.15	TCP	2934 80 → 38864 [PSH, ACK] Seq=563928 Ack=8924 Win=65535 Len=2880 [TCP PDU reassembled in 369]
348	23.084642	10.0.2.15	151.101.130.132	TCP	54 38864 → 80 [ACK] Seq=8924 Ack=563928 Win=65535 Len=0
349	23.084974	10.0.2.15	151.101.130.132	TCP	54 38864 → 80 [ACK] Seq=8924 Ack=568880 Win=65535 Len=0
350	23.085239	151.101.130.132	10.0.2.15	TCP	1514 80 → 38864 [PSH, ACK] Seq=568880 Ack=8924 Win=65535 Len=1440 [TCP PDU reassembled in 369]
351	23.085239	151.101.130.132	10.0.2.15	TCP	934 80 → 38864 [PSH, ACK] Seq=569248 Ack=8924 Win=65535 Len=880 [TCP PDU reassembled in 369]
352	23.085239	151.101.130.132	10.0.2.15	TCP	594 80 → 38864 [PSH, ACK] Seq=569184 Ack=8924 Win=65535 Len=540 [TCP PDU reassembled in 369]
353	23.085240	151.101.130.132	10.0.2.15	TCP	2934 80 → 38864 [PSH, ACK] Seq=569888 Ack=8924 Win=65535 Len=2880 [TCP PDU reassembled in 369]
354	24.085748	151.101.130.132	10.0.2.15	TCP	2934 80 → 38864 [PSH, ACK] Seq=577664 Ack=8924 Win=65535 Len=2880 [TCP PDU reassembled in 369]

Rysunek 6: Ruch grupy P6: cykliczne wysyłanie żądań HTTP DELETE dotyczących tego samego zasobu.

Wstępne metody, takie jak proste przeszukiwanie ciągów znaków, nie przyniosły rezultatów.

### 2.6.1. Identyfikacja Kanału Ukrytego

Szczegółowa inspekcja pakietów HTTP ujawniła systematyczne anomalie w niestandardowych polach nagłówkowych, zaczynających się od prefiksu `X-` (np. `X-Hidden-Data`). Pola te charakteryzowały się użyciem

zmiennej liczby spacji pomiędzy słowami, co jest klasyczną techniką steganografii opartej na białych znakach (ang. *whitespace steganography*). Zidentyfikowano ten mechanizm jako prawdopodobny kanał ukryty, służący do transmisji dodatkowych danych.

### 2.6.2. Ekstrakcja Ukrytych Danych

Do ekstrakcji danych z zidentyfikowanego kanału ukrytego zastosowano metodę binarną opartą na parzystości liczby spacji. Każda sekwencja białych znaków pomiędzy słowami w nagłówkach X- została przetworzona zgodnie z regułą:

- Parzysta liczba spacji odpowiada bitowi 0.
- Nieparzysta liczba spacji odpowiada bitowi 1.

Zastosowanie tej reguły do wszystkich znalezionych nagłówków pozwoliło na zrekonstruowanie binarnego strumienia danych. W wyniku tego procesu pomyślnie wyodrębniono ciąg 463 bajtów, stanowiący prawdopodobny szyfrogram. Pierwsze 16 bajtów szyfrogramu w notacji heksadecymalnej to: 533531A197CCB9CF64B7B11FB2ADD88F.

### 2.6.3.

sectionAnaliza Kryptograficzna Przyjęto hipotezę, że uzyskany ciąg bajtów jest szyfrogramem powstałym w wyniku zastosowania symetrycznego szyfru strumieniowego, najprawdopodobniej szyfru Vigenère'a z operacją XOR. Podjęto próbę odnalezienia klucza deszyfrującego.

### 2.6.4. Testowanie Listy Prawdopodobnych Kluczy

W pierwszej kolejności przeprowadzono zautomatyzowany atak, testując listę kluczy wygenerowanych na podstawie kontekstu zadania. Lista ta (Tabela 3) zawierała słowa pochodzące z nazw plików, treści dramatu oraz technicznych aspektów analizowanych logów. Każdy z kluczy został użyty do operacji XOR na szyfrogramie, a wynik sprawdzono pod kątem zgodności z oczekiwanym początkiem tekstu jawnego („Aby rozpocząć lekturę...”).

Tabela 3: Lista kluczy przetestowanych w ataku.

Kategoria	Przykładowe Klucze
Z nazw plików	stego, yes, pcap, Sofokles, Antygona
Postacie z dramatu	Kreon, Ismena, Haimon, Eurydyka, Edyp
Z treści logów	Debian, curl, example.com, http, dns
Z kontekstu	best, lab, bestlab, p6, studia, BEST

Analiza wykazała, że żaden z powyższych kluczy nie pozwolił na odtworzenie poprawnego tekstu jawnego.

### 2.6.5. Próba Ataku ze Znanym Tekstem Jawnym

Wobec niepowodzenia poprzedniej metody, podjęto próbę ataku z tekstem jawnym (ang. *known-plaintext attack*). Wykorzystano fakt znajomości szyfrogramu oraz prawdopodobnego początku tekstu jawnego do wyliczenia klucza za pomocą operacji  $\text{Klucz} = \text{Szyfrogram} \text{ XOR } \text{TekstJawny}$ . Analiza pierwszych ośmiu bajtów dała następujący fragment klucza: 12 57 48 81 E5 A3 CD BF. Zastosowanie tego 8-bajtowego, powtarzalnego klucza do całego szyfrogramu również nie przyniosło oczekiwanego rezultatu w postaci spójnego tekstu w języku polskim.

### 2.6.6. Wnioski

Zastosowane metody kryptoanalizy, zarówno poprzez testowanie listy prawdopodobnych haseł, jak i poprzez atak z użyciem znanego tekstu jawnego, nie doprowadziły do odtworzenia finalnej wiadomości. Świadczy to o tym, że albo klucz deszyfrujący jest nieoczywisty i pochodzi z innego źródła, albo jedna z pierwotnych hipotez (np. dotycząca dokładnej treści początku tekstu jawnego lub metody szyfrowania) była błędna. Ostateczne złamanie szyfru wymaga dodatkowych prób i kombinacji.



4	0.80048874	192.168.1.22	192.168.1.28	HTTP	454 GET /filestreaming-service/files/a3243ae-852-48c8-9185-e83c5f64e1791c1746487088f2-4848f3-18f4-Qo51Khvrbv7a518u3Zf8KdLgCakabpa25paiBuYSB0YwtpIHBY
5	0.80149597	192.168.1.28	192.168.1.22	TCP	66 8000 → 8000 [ACK] Seq=1 Ack=389 Win=64896 Len=0 Tval=1708124786 Tsc=1512794555
6	0.80112586	192.168.1.28	192.168.1.22	TCP	159 8000 → 48208 [PSH, ACK] Seq=1 Ack=389 Win=64896 Len=0 Tval=1708124788 Tsc=1512794555 [TCP RDU reassembled in 7]
7	0.80134861	192.168.1.28	192.168.1.22	HTTP	66 HTTP/1.0 200 OK
8	0.80116223	192.168.1.22	192.168.1.28	TCP	66 48208 → 8000 [ACK] Seq=389 Ack=94 Win=64256 Len=0 Tval=1512794558 Tsc=1708124788
9	0.80057918	192.168.1.22	192.168.1.28	TCP	66 8000 → 8000 [FIN, ACK] Seq=389 Ack=94 Win=64256 Len=0 Tval=1512794558 Tsc=1708124788
10	0.80114449	192.168.1.28	192.168.1.22	TCP	66 8000 → 8000 [ACK] Seq=97 Ack=92 Win=64896 Len=0 Tval=1708124790 Tsc=1512794559
11	0.71034424	192.168.1.22	192.168.1.28	TCP	74 58572 → 8000 [FIN] Seq=0 Ack=94048 Len=0 PSH=1448 SACK_PERR Tval=1512794560 Tsc=0 Win=128
12	0.72237052	192.168.1.28	192.168.1.22	TCP	74 8000 → 58572 [PSH, ACK] Seq=0 Ack=94 Win=63188 Len=0 PSH=1448 SACK_PERR Tval=1708124807 Tsc=1512795266 Win=128
13	0.71037824	192.168.1.22	192.168.1.28	TCP	66 58572 → 8000 [ACK] Seq=1 Ack=1 Win=64256 Len=0 Tval=1512795267 Tsc=1708124807
14	0.71037806	192.168.1.22	192.168.1.28	HTTP	454 GET /filestreaming-service/files/935a208e-4845-47c2-8a5a-f8f2c23f28f4791c1746487088f2-4848f3-18f4-Qo51Khvrbv7a518u3Zf8KdLgCakabpa25paiBuYSB0YwtpIHBY
15	0.71374349	192.168.1.28	192.168.1.22	TCP	66 8000 → 58572 [ACK] Seq=1 Ack=391 Win=64896 Len=0 Tval=1708124808 Tsc=1512795269
16	0.71374353	192.168.1.28	192.168.1.22	TCP	159 8000 → 58572 [PSH, ACK] Seq=1 Ack=391 Win=64896 Len=0 Tval=1708124808 Tsc=1512795269 [TCP RDU reassembled in 18]
17	0.71374353	192.168.1.22	192.168.1.28	TCP	66 58572 → 8000 [ACK] Seq=391 Ack=94 Win=64256 Len=0 Tval=1512795272 Tsc=1708124808
18	0.71374359	192.168.1.28	192.168.1.22	HTTP	66 HTTP/1.0 200 OK
19	0.71377694	192.168.1.22	192.168.1.28	TCP	66 58572 → 8000 [FIN, ACK] Seq=391 Ack=94 Win=64256 Len=0 Tval=1512795273 Tsc=1708124808
20	0.71377648	192.168.1.28	192.168.1.22	TCP	66 8000 → 58572 [ACK] Seq=97 Ack=92 Win=64896 Len=0 Tval=1708124804 Tsc=1512795273
21	1.95581795	192.168.1.22	192.168.1.28	TCP	74 58508 → 8000 [FIN] Seq=0 Ack=94048 Len=0 PSH=1448 SACK_PERR Tval=1512795284 Tsc=0 Win=128
22	1.95581795	192.168.1.22	192.168.1.28	TCP	74 8000 → 58508 [FIN, ACK] Seq=0 Ack=94 Win=63188 Len=0 PSH=1448 SACK_PERR Tval=1708124804 Tsc=1512795284
23	1.95621716	192.168.1.22	192.168.1.28	TCP	66 58508 → 8000 [ACK] Seq=1 Ack=1 Win=64256 Len=0 Tval=1512795285 Tsc=1708124804
24	1.95621716	192.168.1.22	192.168.1.28	HTTP	454 GET /filestreaming-service/files/ef451e-418a-4a8a-801a-581dbb644da791c1746487088f2-4848f3-18f4-Qo51Khvrbv7a518u3Zf8KdLgCakabpa25paiBuYSB0YwtpIHBY
25	1.95786783	192.168.1.28	192.168.1.22	TCP	66 8000 → 58508 [ACK] Seq=1 Ack=389 Win=64896 Len=0 Tval=1708124802 Tsc=1512795312
26	1.95872248	192.168.1.28	192.168.1.22	TCP	159 8000 → 58508 [PSH, ACK] Seq=1 Ack=389 Win=64896 Len=0 Tval=1708124743 Tsc=1512795311 [TCP RDU reassembled in 27]
27	1.95872248	192.168.1.28	192.168.1.22	HTTP	66 HTTP/1.0 200 OK
28	1.95887779	192.168.1.22	192.168.1.28	TCP	66 58508 → 8000 [ACK] Seq=389 Ack=94 Win=64256 Len=0 Tval=1512795313 Tsc=1708124743
29	1.95887779	192.168.1.22	192.168.1.28	TCP	66 58508 → 8000 [FIN, ACK] Seq=389 Ack=94 Win=64256 Len=0 Tval=1512795313 Tsc=1708124743
30	1.96151794	192.168.1.28	192.168.1.22	TCP	66 8000 → 58508 [ACK] Seq=97 Ack=390 Win=64896 Len=0 Tval=1708124746 Tsc=1512795313
31	1.96231249	192.168.1.22	192.168.1.28	TCP	74 58508 → 8000 [ACK] Seq=0 Ack=94048 Len=0 PSH=1448 SACK_PERR Tval=1512795317 Tsc=0 Win=128
32	1.96231249	192.168.1.22	192.168.1.28	TCP	74 8000 → 58508 [FIN, ACK] Seq=0 Ack=94 Win=63188 Len=0 PSH=1448 SACK_PERR Tval=1708124746 Tsc=1512795317
33	1.96231249	192.168.1.22	192.168.1.28	TCP	66 58508 → 8000 [ACK] Seq=1 Ack=1 Win=64256 Len=0 Tval=1512795317 Tsc=1708124746
34	1.96231249	192.168.1.22	192.168.1.28	HTTP	454 GET /filestreaming-service/files/4ea5b02-38d7-41a2-b0b8-83edcf99ccf891c1746487088f2-4848f3-18f4-Qo51Khvrbv7a518u3Zf8KdLgCakabpa25paiBuYSB0YwtpIHBY
35	1.96231249	192.168.1.28	192.168.1.22	TCP	66 8000 → 58508 [ACK] Seq=1 Ack=391 Win=64896 Len=0 Tval=1708124806 Tsc=1512795317
36	1.96231249	192.168.1.28	192.168.1.22	TCP	159 8000 → 58508 [PSH, ACK] Seq=1 Ack=391 Win=64896 Len=0 Tval=1708124806 Tsc=1512795317 [TCP RDU reassembled in 37]
37	1.96231249	192.168.1.28	192.168.1.22	HTTP	66 HTTP/1.0 200 OK
38	1.97460404	192.168.1.22	192.168.1.28	TCP	66 58508 → 8000 [FIN, ACK] Seq=389 Ack=94 Win=64256 Len=0 Tval=1512795318 Tsc=1708124806

Rysunek 7: Ruch grupy P7: sekwencja pomyślnych żądań HTTP GET pobierających dane z serwera.

## 2.7. P7

### 2.7.1. Identyfikacja Kanału i Metody Kodowania

Wstępna inspekcja pakietów ujawniła anomalię w zapytaniach HTTP GET. W strukturze URI tych zapytań zidentyfikowano stały zestaw parametrów, z których jeden, oznaczony jako P4, zawierał długie ciągi znaków o wysokiej entropii. Zestaw znaków (A-Z, a-z, 0-9, +, /) nasunął jednoznaczną hipotezę o zastosowaniu kodowania Base64.

Hipotezę zweryfikowano poprzez programistyczną ekstrakcję wartości parametru P4 ze wszystkich pakietów w porządku chronologicznym. Połączenie tych fragmentów i próba zdekodowania uzyskanego ciągu Base64 zakończyła się sukcesem. Odkodowany tekst był zapisany w kodowaniu **windows-1250**.

### 2.7.2. Przykłady Zakamuflowanych Danych

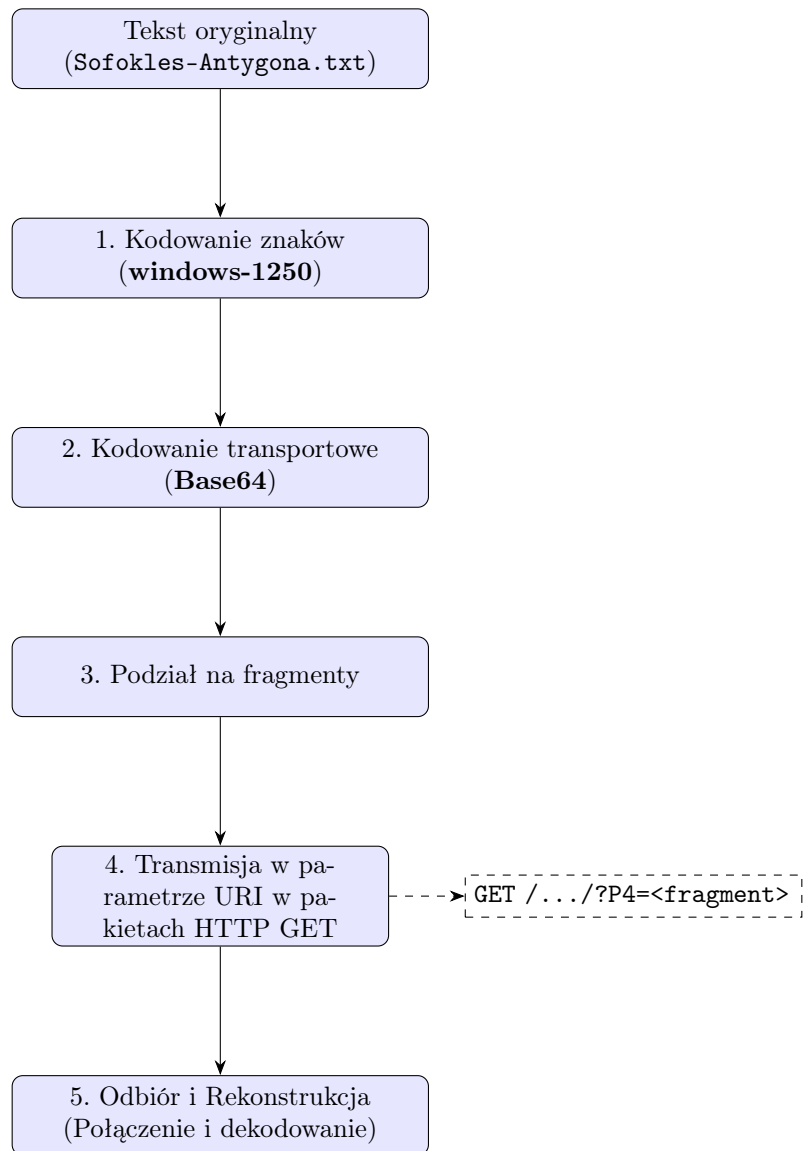
Poniższa tabela przedstawia wybrane pakiety z ukrytymi danymi, ich zakodowaną postać oraz zdekodowany fragment tekstu.

Nr Ramki	Wartość Parametru P4 (Base64)	Odszyfrowany Fragment Tekstu
4	QWJ5IHJvenBvY3q55iBsZWt0dXLq	Aby rozpocząć lekturę
4	LCAKa2xpa25paiBuYSB0YwtpIHBY	, \nkliknij na taki pr
4	enljaXNrIAprdPNyeSBkYSBj	zycisk \nktóry da c
9	aSBwZbNueSBkb3N06nAgZG8gc3Bpc3UgdHJlnGNpIGtzabm	i pełny dostęp do spisu tre- ści książki
9	a2kuIAoKSmWcbGkgY2hjZXN6IHVvs7lj	żki. \n\nJeśli chcesz połącz

Tabela 4: Tabela z przykładami zdekodowanych fragmentów z parametru P4.

### 2.7.3. Schemat Metody Steganograficznej

Zidentyfikowana metoda polegała na wieloetapowym przetwarzaniu danych przed ich transmisją w sieci, co ilustruje poniższy schemat.



Rysunek 8: Schemat zidentyfikowanej metody steganograficznej dla grupy P7.